TEMA D'ANNO

ESAME DI MISURE ELETTRONICHE
CORSO DI LAUREA IN ING. ELETTRONICA V.O.

Progetto, realizzazione, collaudo e utilizzo di una matrice di switching automatica, implementata con microcontrollore, conforme allo standard industriale SCPI (IEEE-488.2)

DOCENTI:
Prof. Ing. Nicola GIAQUINTO
Dott. Ing. Francesco ADAMO

STUDENTE:
Paolo SANCONO
Matr. 511697 B

ANNO ACCADEMICO 2005

“Progetto, realizzazione, collaudo e utilizzo

di una matrice di switching automatica,

implementata con microcontrollore,

conforme allo standard industriale SCPI (IEEE-488.2)” V 1.1

L’autore non si assume nessuna responsabilità per danni diretti o indiretti derivanti dall’uso delle informazioni riportate in questo documento. Chi decide di realizzare il circuito illustrato, o utilizzare in altro modo le altre informazioni presenti nel documento, si assume tutte le responsabilità.

Ringraziamenti

Un sentito ringraziamento ai docenti del corso di Misure Elettroniche, prof. ing. Nicola Giaquinto e dott. ing. Francesco Adamo, per le chiare spiegazioni fornite su ogni aspetto della disciplina, per la straordinaria competenza in materia, per la conoscenza delle più moderne innovazioni tecnologiche disponibili sul mercato, e per il giusto equilibrio didattico tra approfondimenti teorici e prove pratiche.

Seguire il corso ha definitivamente ampliato le mie conoscenze e fornitomi strumenti concreti di misura automatica di grandezze fisiche, elettriche e non, valutazione dell'incertezza, condizionamento dei segnali provenienti da sensori e trasduttori, algoritmi di analisi ed elaborazione automatica dei dati, strumenti di controllo automatico e remoto, indispensabili nelle attività ingegneristiche.

Un sentito ringraziamento anche per la disponibilità dimostrata in studio e in laboratorio dai docenti per appuntamenti fuori dall'orario di lezione: alcuni studenti si sono letteralmente accampati in laboratorio per concludere i propri esperimenti, abusando della pazienza dei docenti che è sembrata senza fine.

Indice

Contenuti

In questo tema d’anno vengono descritte tutte le fasi che hanno portato alla realizzazione di una matrice di switching da utilizzare in un laboratorio ATE.

Nel **Capitolo 1, Introduzione**, dopo aver descritto sommariamente in cosa consiste un laboratorio ATE e alcuni dei problemi che si possono incontrare durante le sperimentazioni, viene presentato il sistema oggetto del tema d’anno, che risolve alcune problematiche tipiche, con un costo inferiore di almeno un ordine di grandezza ai prodotti commerciali equivalenti. Dall’idea generale del circuito, si passa ad una descrizione più dettagliata, che mostra cosa il sistema è in grado di fare. Si è riusciti, nell’introduzione, a dare una descrizione accurata del sistema, evitando, però, di soffermarsi sui dettagli implementativi.

Nel **capitolo 2, Hardware**, viene trattato diffusamente della realizzazione fisica del sistema. Inizialmente vengono descritte le sperimentazioni sulle diverse sotto-sezioni del circuito, effettuate prima di affrontare la realizzazione del sistema completo. Viene descritta sommariamente l’architettura del PIC16F628, il microcontrollore scelto, degli altri chip utilizzati nel progetto, e del programmatore autocostruito, il cui progetto è di Fiser, che lo ha reso disponibile sul suo sito Web. Nello stesso capitolo è presente una breve panoramica sulla comunicazione seriale.
Una volta chiarito il funzionamento dei sotto-sistemi, si presenta schema elettrico e sbroglio su basetta del circuito definitivo, con ampio spazio destinato alla descrizione del funzionamento elettrico, alle scelte di sbroglio, e ai procedimenti chimici e meccanici utilizzati per incidere e assemblare le basette. Viene presentato il software *EagleCad* e lo script *pshouse-drill-aid*, realizzato per il tema d’anno, utilizzati per il disegno delle basette.
Nello stesso capitolo si parla anche del collaudo del circuito, delle misure di assorbimento, test di riscaldamento, test di comunicazione su cavi lunghi, burn-in test, self-test, dimensioni fisiche, analisi dei costi iniziale e consuntivo finale.

Questo progetto era impossibile da portare a termine senza una adeguata progettazione. Nel **capitolo 3, Progettazione e analisi del protocollo**, vengono descritti gli strumenti di ausilio all’organizzazione delle idee e alla progettazione vera e propria del circuito e del firmware. Viene sottolineata l’importanza di un glossario dei termini, e di diversi documenti testuali di

documentazione interna (per il progettista stesso) ed esterna (per il cliente/utilizzatore), indispensabili in ogni fase di progettazione e documentazione finale.
Nello stesso capitolo viene descritto succintamente il protocollo SCPI (standard industriale IEEE-488.2), l'analisi di questo protocollo effettuata, le scelte operate per i comandi da implementare, e il sistema di segnalazione di errori hardware (il circuito è in grado di effettuare un auto-test) e software progettato *ad hoc*, che si distanzia dallo standard, al quale si è invece rimasti fedeli per quel che riguarda la sintassi dei comandi.
Corredano il capitolo le tabelle di riferimento rapido per l'utilizzatore della macchina con la sintassi dei comandi e i messaggi di errore generati.

Il **capitolo 4** è dedicato a tutto ciò che concerne il **firmware**, a cominciare dalle differenze tra un software su un PC e un firmware su un microcontrollore, e come queste differenze portino a procedure di messa a punto (debug) completamente differenti, per continuare con l'estensione dell'assembler realizzata con la raccolta di macro *psifthen,* di mia ideazione, in grado di fornire il supporto per strutture di controllo di flusso avanzate: costrutti if-then-else, cicli for, while, repeat, eccetera, in un linguaggio che ne è privo. Viene descritto il BIOS implementato sul microcontrollore, ovvero il sottosistema di accesso all'hardware, le routine assembler per la gestione di stringhe a lunghezza variabile, la routine di auto-test dell'hardware, il sotto-sistema di gestione degli errori e della comunicazione seriale, per finire con la descrizione del ciclo principale del programma e del parser, lo strumento software in grado di riconoscere i costrutti del protocollo SCPI, di segnalare errori di sintassi, e di eseguire i comandi validi riconosciuti.

Nel **capitolo 5** vengono mostrati **esempi d'uso** della macchina con la scrittura diretta dei comandi tramite il *Terminale di Windows*, con uno strumento di calcolo scientifico in grado di acquisire dati da strumentazione automatica, come il *Matlab*, e con un ambiente di sviluppo grafico specifico per l'analisi dei dati e l'utilizzo in remoto di strumentazione, dotato di GUI accattivanti, come il *Labview*. Di ciascun esempio vengono forniti tutti i dettagli, e vengono dati suggerimenti per chi intenda utilizzare la macchina con altri linguaggi di programmazione.

Completano il tema d'anno il **capitolo 6, Conclusioni**, dove si inserisce il progetto in un contesto di più ampio respiro e si ipotizzano prospettive future di sviluppo fino alla realizzazione di un intero laboratorio utilizzabile in remoto tramite Web, le **4 Appendici**, con

annotazioni pratiche importanti per l’utilizzo del circuito, i listati integrali del firmware e degli script Matlab, il testo delle licenze GNU FDL e GNU GPL, con le quali sono stati distribuiti l’intero lavoro e i codici sorgente dei programmi realizzati, e infine le **fonti delle figure,** l’elenco dei principali **software utilizzati** e una **bibliografia e webliografia** ragionate.

Capitolo 1 - Introduzione

Il corso di "Misure Elettroniche" del Politecnico di Bari, tenuto dai docenti prof. ing. N. Giaquinto e dott. ing. F. Adamo, ha, tra gli argomenti centrali, l'utilizzo di strumentazione ATE[1] per l'effettuazione di alcune serie di misure in automatico, sotto la supervisione di un programma di controllo scritto sul PC.

Un laboratorio ATE

Nella totalità dei casi, occorre realizzare fisicamente il collegamento tra i dispositivi o i circuiti in test e la strumentazione di misura automatica, posto che il collegamento tra quest'ultima e il PC sia già stato effettuato. In molti casi di interesse pratico, la topologia del circuito va modificata per concludere la serie di misure di interesse.

Un semplice esempio: per determinare sperimentalmente la caratteristica di un dispositivo elettronico con l'utilizzo di un alimentatore e un multimetro da banco programmabili, è necessario invertire l'alimentazione del dispositivo poiché gli alimentatori difficilmente riescono a fornire tensioni positive e negative dalla stessa coppia di morsetti.

Con questo tipo di inconveniente, si perdono molti dei vantaggi derivanti dalla possibilità di eseguire in automatico una serie di misure, poiché è ancora necessario l'intervento manuale per modificare la topologia del circuito. Le conseguenze sono ancora peggiori nei casi in cui interessa la ripetibilità delle misure in condizioni particolari. Cablando nuovamente il circuito,

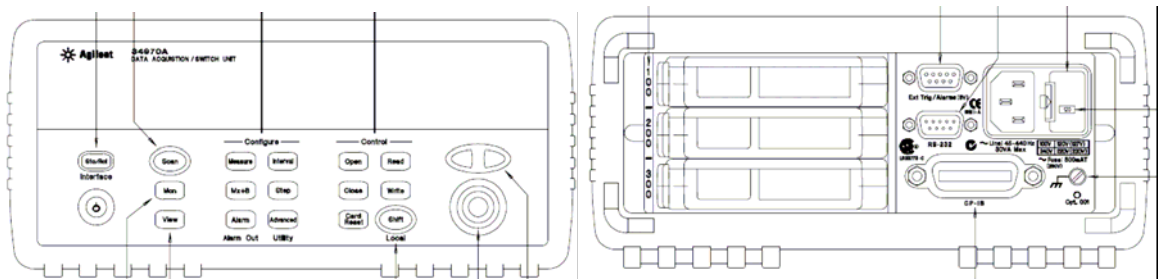
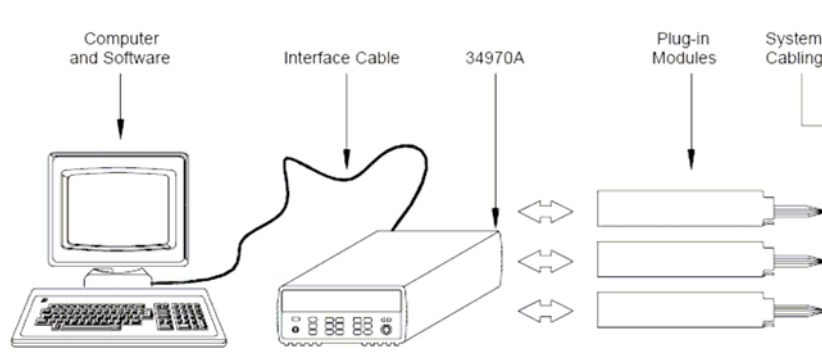
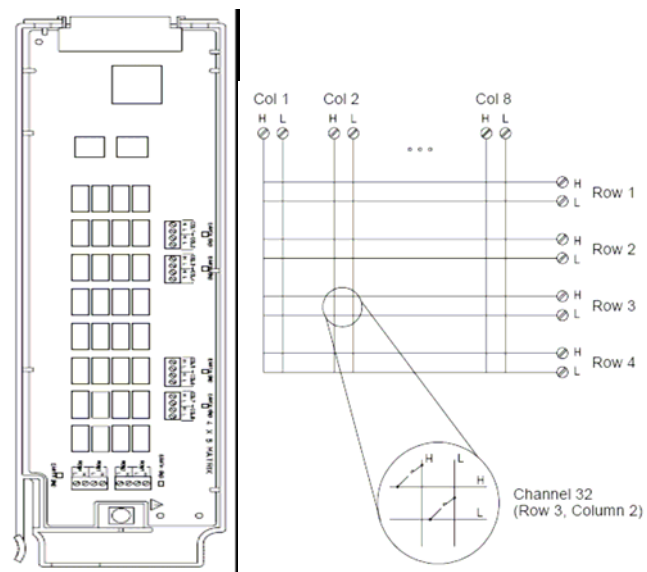
possono variare le resistenze di contatto delle interconnessioni e le misure possono svolgersi in condizioni operative differenti.
Il sistema descritto in questo tema d'anno, una matrice di switching modulare, conforme al protocollo SCPI (Standard industriale IEEE-488.2), viene proprio incontro a questa esigenza. Nei successivi paragrafi dell'introduzione vedremo come è nata, in cosa consiste e come si è sviluppata l'idea, fino alla formulazione di specifiche generali che hanno consentito le successive fasi di progetto, realizzazione, collaudo e utilizzo della macchina.

Specifiche iniziali

Idea

Per esercizio didattico, e per potenziare le funzionalità del nostro laboratorio[2], si è realizzato un circuito comprendente 16 relais[3] SPST organizzati in una matrice 4x4 in grado di collegare uno qualsiasi di 4 ingressi ad una qualsiasi di 4 uscite. E' possibile eccitare anche più relais contemporaneamente in modo da realizzare collegamenti multipli.
Il circuito possiede anche la logica di controllo necessaria per interfacciarsi con la porta seriale RS232 di un PC. Il protocollo di comunicazione da usare è lo standard SCPI (Standard Commands for Programmable Instruments) IEEE-488.2, nella versione dello strumento da laboratorio Agilent 34970A Data Acquisition/Switch Unit, prodotto commerciale preso a modello generale durante il lavoro.
Esistono kit che consentono di trasformare uno strumento che operi su seriale in uno operante in GPIB, quindi la scelta fatta non è limitante, qualora si intendesse usare l'altro tipo di bus di connessione. La porta seriale resta la scelta più economica ed è l'unica porta di comunicazione realmente onnipresente, se non di serie, almeno come opzione su qualunque sistema elettronico in grado di scambiare dati.

---

[1] ATE = Automated Test Equipment: Strumentazione di acquisizione dati automatica.
[2] Laboratorio di Misure Elettroniche al Dipartimento di Elettrotecnica ed Elettronica (DEE) del Politecnico di Bari.
[3] Nella versione finale, il circuito è modulare ed è scalabile fino a 64 relais.

Pannello frontale e posteriore dell’Agilent 34970A.

Topologia di collegamento a stella dell’Agilent 34970A a moduli di switch di espansione

34904A: modulo di espansione con matrice di switch 4x8 per il 34970A

Il prodotto Agilent è costituito da una unità principale, da collegare al PC o utilizzare dal pannello frontale, dotata di 3 slot in cui inserire altrettanti moduli, secondo una topologia di collegamento a stella. I moduli contengono le matrici o i multiplexer di switch, o altre funzionalità non di interesse per questo progetto.

Il circuito è utilissimo in un laboratorio di misure automatiche, poiché consente di collegare automaticamente gli strumenti di misura a differenti circuiti in prova, di realizzare

automaticamente misure che richiedono normalmente un intervento manuale, di creare automaticamente da programma veri e propri circuiti connettendo al volo tra loro componenti o interi stadi già pronti. Il circuito è una alternativa molto economica[4] ai prodotti commerciali, ideale per un uso didattico, con l'ulteriore vantaggio, non da poco conto in tale contesto, di poter conoscere tutto dell'architettura interna per attività di studio, di miglioramento o di riparazione.

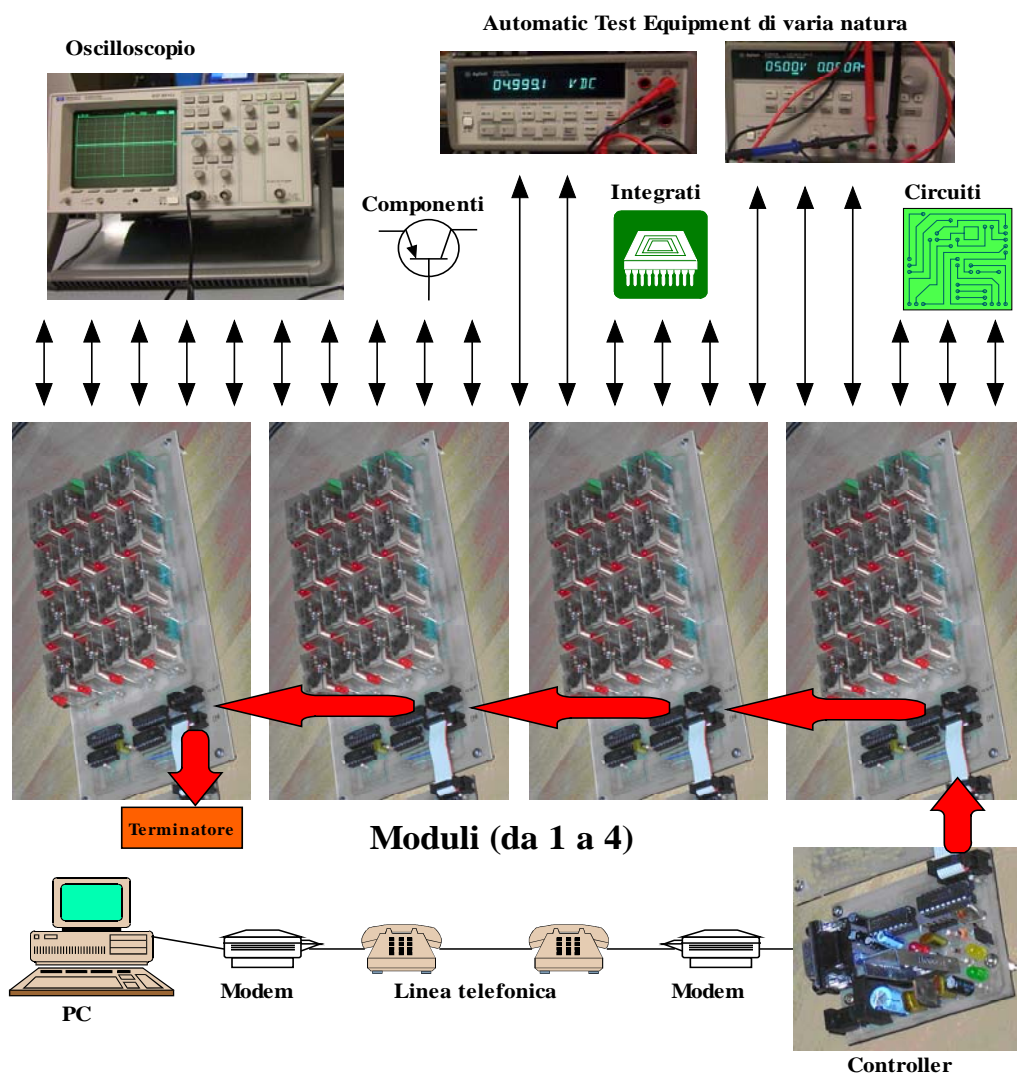
Descrizione sommaria

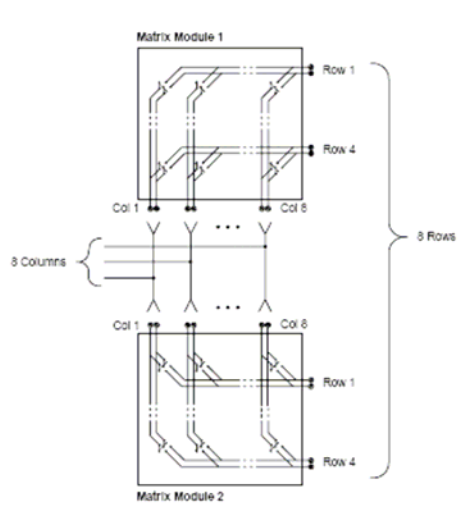
Dalla idea generale, si è passati ad una descrizione sommaria del progetto, con l'intento di determinare quali potenze di calcolo ed elettriche fossero in gioco, su quali e quanti circuiti integrati orientarsi, e su quale tecnologia utilizzare per realizzare il circuito.

L'idea proposta prevedeva di poter realizzare il circuito come blocco unico, anche su piastra millefori, e senza led di segnalazione. Sviluppando il progetto, si è invece pensata una architettura scalabile in cui il **controller** (basetta comprendente stadio di alimentazione, un microcontrollore, un adattatore di livello e la presa seriale), comunicava attraverso un **bus di sistema** (cavo piatto a 10 fili) con uno o più **moduli** (basette comprendenti shift-register con latch, darlington di potenza e relais). Il numero massimo di moduli gestibile è 4, per un totale di 64 relais. La topologia del collegamento tra moduli è in cascata[5], con terminatore sull'ultimo. I diversi moduli possono essere sistemati uno accanto all'altro sul piano di lavoro, oppure impilati uno sopra l'altro con distanziatori opportuni, se è necessario risparmiare superficie di appoggio.

Il sistema consente di collegare dinamicamente tra loro strumenti da laboratorio, ad esempio oscilloscopi, generatori di funzione, multimetri da banco, alimentatori programmabili, componenti in prova, sensori integrati di temperatura, luminosità o quant'altro, interi circuiti in prova o di condizionamento del segnale. E' anche possibile montare dinamicamente, da programma, interi circuiti connettendo tra loro componenti diversi, montati una volta per tutte sul sistema. I vantaggi, come si vede, sono notevoli.

---

[4] I costi di acquisto e di riparazione sono di un ordine di grandezza inferiori.
[5] Ogni modulo contiene un connettore di ingresso e uno di uscita per il bus.

Il sistema *Relais Matrix* completo: PC, modem e linea telefonica, controller, 4 moduli per 64 relais, strumentazione ATE, componenti, sensori integrati, circuiti in test o di condizionamento del segnale, tutto collegabile dinamicamente da programma di controllo remoto, tramite le matrici di switching.

Se si ha bisogno di matrici di switching più ampie di 4x4, si possono combinare tra loro più moduli, collegando righe o colonne omologhe, come esemplificato nel diagramma seguente, relativo alla macchina Agilent di riferimento.

Unire due matrici 4x8 a formare una matrice 8x8

Si è pensato di inserire sul controller un led di segnalazione e un buzzer piezoelettrico, in modo da segnalare con un beep la ricezione di comandi con sintassi non valida dal PC, così come la maggioranza degli strumenti di acquisizione dati automatici. Il circuito prevede tre alimentazioni: 5V per la logica, duale +10V -10V per l'interfaccia seriale, +12V per i relais. Sono stati inseriti direttamente sul circuito due led di segnalazione alimentazione presente sulle linee 5V e 12V, in modo da avere subito indicazione visiva di malfunzionamenti critici nello stadio di alimentazione prima di cercare problemi altrove. La tensione duale +10V -10V viene invece generata direttamente dallo stesso chip (Maxim MAX232) che si occupa di adattare i livelli TTL 0/5V in livelli RS232 +10V/-10V e non è stato inserito il led di segnalazione per non caricare gli stadi duplicatore ed invertitore dell'integrato.

Il circuito doveva per forza prevedere un microcontrollore, e ci si è orientati verso i PIC della Microchip, poiché sono disponibili numerose risorse gratuite e una fervente comunità di appassionati su Internet a cui rivolgersi in caso di difficoltà. Inoltre la serie di prodotti a catalogo è nutrita, la maggioranza di questi sono pin-2-pin compatibili l'uno con l'altro, per consentire di migrare, in caso di necessità, verso uno più potente senza dover ridisegnare il circuito; quasi tutti i modelli sono disponibili in package through hole PDIP (alla portata del saldatore manuale) e la rete di distribuzione è ben sviluppata consentendo l'acquisto anche in un singolo esemplare a prezzi contenuti dalla quasi totalità dei rivenditori di componenti elettronici italiani. Esistono altre famiglie di microcontrollori più potenti e versatili o più facili da programmare, ma è più difficile trovare documentazione e disponibilità altrettanto esauriente e capillare.

Descrizione più dettagliata

Dalla descrizione sommaria si è dovuti passare ad una descrizione più dettagliata, che consentisse di stilare un preventivo dei costi e iniziare il progetto della parte hardware del circuito. Per facilitare la discussione su queste questioni col docente, che simulava il cliente del sistema, è stata preparata una lista di domande sulle specifiche del circuito, con una o due proposte per le risposte, e un preventivo di massima aperto, dal quale aggiungere e togliere componenti in caso di variazioni di progetto, o disponibilità o meno del materiale già nel laboratorio dell'Università. Grazie a questo lavoro fatto a casa, l'elaborazione di specifiche più dettagliate col docente è stata piuttosto rapida.

Per **l'alimentazione** si è pensato ad un alimentatore dedicato, poiché è snervante dover cercare un alimentatore in DC disponibile ogni volta che si vuole utilizzare un circuito. Inizialmente si pensava di alimentare il circuito direttamente a tensione di rete 240V AC, e di montare lo stadio di alimentazione direttamente sul circuito. Per risparmiare sui costi del trasformatore, e per evitare i potenziali pericoli derivanti dall'avere piste a tensione di rete, potenzialmente letali, sulla basetta, si è invece deciso di utilizzare un alimentatore già disponibile in laboratorio, da dedicare permanentemente al circuito: un *wall-cube*[6] con uscita 12V AC 1.5A, che sembrava proprio di potenza adeguata per gli assorbimenti previsti del circuito (100 mA per relais, 16 relais: 1.6A @12VDc). Lo stadio di stabilizzazione sarebbe stato comunque localizzato sulla basetta del controller, e sarebbe stato in grado di accettare un range piuttosto ampio di tensioni di ingresso in modo da facilitare la sostituzione dell'alimentatore in caso di rottura.

La complessità del circuito consigliava, se non obbligava, a realizzare una **basetta incisa** almeno **monofaccia**, anziché utilizzare le basette millefori come si pensava inizialmente.

Avendo deciso di separare su due basette diverse il controller dai moduli, e avendo deciso di prevedere la possibilità di collegare un numero variabile di moduli al controller, era necessario prevedere un minimo di logica di controllo anche sui moduli e un **bus di sistema** con possibilità di comunicazione dei dati, anche se solo monodirezionale dal controller ai moduli. Era desiderabile la possibilità di poter realizzare la rilevazione automatica del numero di moduli connessi. Entrambi i problemi sono stati risolti semplicemente, economicamente ed elegantemente con degli shift-register dotati di latch, da collegare in cascata in numero

[6] Termine sintetico inglese per indicare gli "scatolotti neri con spina" che si inseriscono direttamente in una presa di corrente e forniscono bassa tensione AC o DC per piccoli apparecchi elettronici. Per tutta la relazione, si è preferito utilizzare un termine inglese quando è più sintetico e/o più preciso del corrispondente italiano.

arbitrario. Dal controller escono i segnali di data, clock e store[7] che finiscono sul primo shift-register della prima basetta; l'ultimo bit di ciascuno shift-register passa all'ingresso data del successivo, così per tutti gli shift-register su una basetta, e analogamente dall'ultimo di una basetta al primo della basetta successiva, finché sull'ultimo modulo, inserendo un ponticello di terminazione, i dati vengono riportati ad un ingresso di feedback del controller, che può così, inviando un pattern di bit e contando dopo quanti cicli di clock torna indietro, determinare quante schede sono state collegate o rilevare l'assenza del terminatore (catena aperta). Allo stesso modo è possibile rilevare guasti sulla quasi totalità dei componenti presenti, per cui si è pensato di realizzare un circuito in grado di effettuare l'autotest.

L'idea di base è che ogni modulo sia identico agli altri, ma il sistema si presta, con piccole o nessuna modifica, alla possibilità di collegare moduli con funzioni differenti, purché abbiano a bordo uno shift-register di 16 bit. Si può quindi pensare a stadi di relais con collegamento a multiplexer anziché a matrice, oppure che rendano disponibili semplicemente i terminali dei loro interruttori per un uso qualsiasi.

Per l'**interconnessione** tra controller e moduli, si è deciso di usare i connettori più economici che garantissero una affidabilità e resistenza meccanica minima, facilità di inserimento/disinserimento e densità tale da consentire una agevole saldatura. Ci si è orientati verso i cavi piatti a 10 fili con connettori a grimpare, con relativi header da circuito stampato. E' lo stesso tipo di connettori usato nel bus IDE dei PC, la differenza è che si è utilizzata la versione a 10 fili, anziché a 40 fili. La versione a 10 fili è molto popolare su circuiti di vario tipo, ad esempio si usa lo stesso connettore, col nome JTAG, per programmare diverse famiglie di FPGA.

[7] L'esatta funzione di ciascuno di questi segnali verrà chiarita meglio in seguito.

I connettori a grimpare utilizzati per il bus di sistema, e un esempio di cavo piatto già montato

I connettori presentano uno smusso da un lato che impedisce di inserirli capovolti. E' piuttosto facile realizzare cavetti della lunghezza desiderata: basta tagliare il cavo piatto a misura ed esercitare pressione sui connettori, che perforeranno la guaina del cavo realizzando automaticamente le connessioni.. Il passo tra i pin è un decimo di pollice, tranquillamente alla portata del saldatore manuale, e tale da consentire di far passare piste da 16 mils tra un piedino e l'altro, cosa fondamentale per uno sbroglio monofaccia.

I segnali da far transitare sul bus sono:

- Riferimento di massa per alimentazione;
- Alimentazione per la logica +5V;
- Alimentazione per i relais +12V;
- Segnali di Data, Clock, Store in logica TTL 0/5V;
- Segnale di feedback/terminazione in logica TTL 0/5V.

Avendo solo 7 segnali e 10 fili a disposizione, si è scelto di raddoppiare l'alimentazione a 12V e di triplicare il ritorno di massa. I connettori e i cavi usati infatti sono adatti per correnti di 1A per filo, mentre l'alimentazione a 12V può superare tale valore, rendendo necessario dedicargli un altro filo in parallelo. Sulla necessità di dedicare piste e cavi quanto più grossi possibile al ritorno di massa, e di bypassare le alimentazioni con condensatori non appena la lunghezza delle piste supera la decina di centimetri o anche meno, non spenderò molte parole considerandolo un fatto acquisito. Dirò qui solo che il rame che costituisce le piste ha la sua resistività, e quando le piste diventano più lunghe e ci scorre molta corrente il potenziale elettrico non è più lo stesso da una parte all'altra. Avere un riferimento di massa non certo sulla basetta può spostare le soglie degli stadi di ingresso dei circuiti digitali, che possono

leggere in modo errato i segnali digitali in ingresso, quindi è importante che sulle piste di massa possa scorrere liberamente quanta più corrente possibile per garantire un potenziale uniforme.

Lo standard *de facto* **RS-232** prevede due tipi di macchine: DTE e DCE[8], che presentano collegamenti invertiti l'una rispetto all'altra per quanto riguarda le linee di trasmissione e ricezione dati. La strumentazione da laboratorio è di solito cablata come DTE, e si è deciso di seguire la stessa regola, montando un connettore SUB9 maschio sulla basetta, cablato come DTE. E' necessario quindi un cavo *null-modem*, ovvero con i fili invertiti[9], per collegare il controller al PC. Con questa soluzione dovrebbe essere possibile utilizzare la scheda in remoto, attraverso la linea telefonica, collegandola ad un modem configurato in *auto-answer*.

Per quanto riguarda il **buzzer piezo-elettrico**, si è scelto il tipo con oscillatore integrato, vale a dire che basta alimentarlo per ottenerne un suono. La corrente consumata dal buzzer è dell'ordine di 15mA, e lo stesso può quindi essere collegato direttamente ad un piedino di uscita del microcontrollore, in grado di erogare o assorbire 20mA da ciascuna uscita digitale: non è stato quindi necessario inserire un transistor per pilotare il buzzer.

Con le informazioni raccolte è stato stilato un **preventivo di massima**, che ometto per brevità, che poi si è rilevato grosso modo esatto e praticamente identico al consuntivo finale, che verrà presentato in seguito.

---

[8] DTE = Data Terminal Equipment, apparecchio che produce e utilizza i dati, tipicamente un PC. DCE = Data Communication Equipment, apparecchio che trasporta i dati verso una destinazione remota, tipicamente un Modem.

[9] L'indicazione su come collegare i pin per costruire un cavo null-modem verrà data in seguito.

Capitolo 2 - Hardware

Sperimentazione

Non si può pretendere di progettare e costruire da zero un circuito complesso e sperare che funzioni al primo colpo. Specie se il circuito è composto da più parti e le diverse parti devono lavorare assieme, è più produttivo progettare, simulare e prototipizzare le parti separatamente, acquisire esperienza con ciascuna, prima di disegnare il circuito finale. Anche così saranno necessari piccoli aggiustamenti sul circuito con tecniche *cut & wire*[10]. Queste tecniche vanno ovviamente bene per i prototipi, non per la produzione, e per piccole correzioni: se il progetto è completamente sbagliato, occorrerà disegnare e realizzare una nuova basetta: per questo si cerca di limitare al minimo la necessità di ricorrere a queste tecniche con una progettazione attenta, e accumulando esperienza su sotto-sezioni del circuito prima di montarlo tutto assieme.

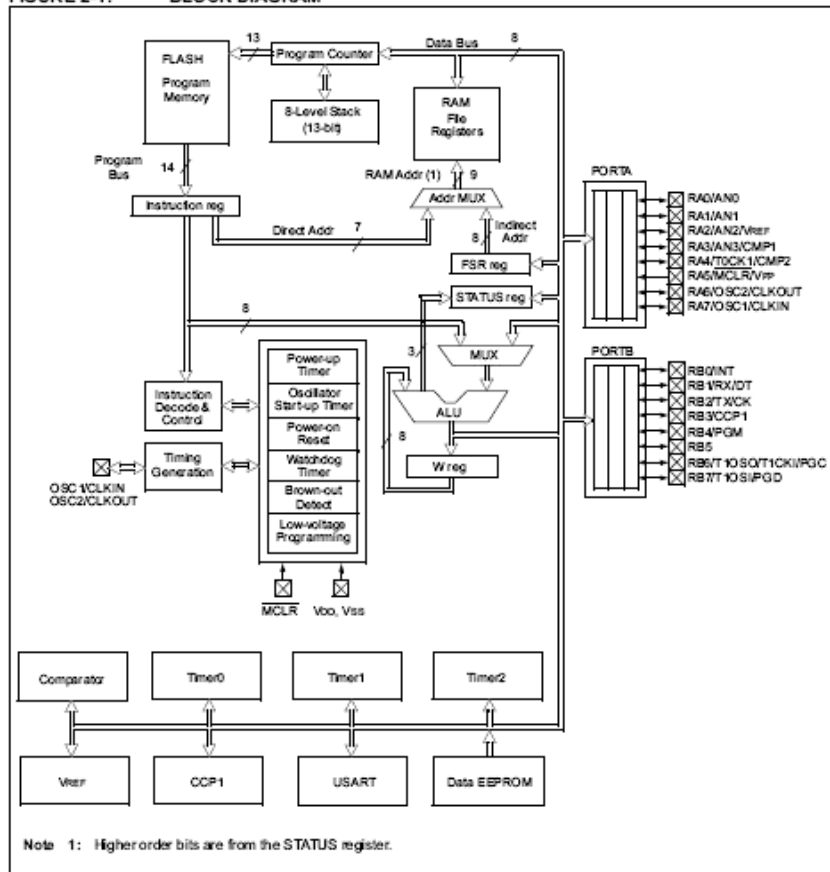
Fiser's programmer

Il microcontrollore scelto

Il microcontrollore scelto per il progetto è stato il **PIC16F628**. Queste sono le caratteristiche salienti di questo microcontrollore: tecnologia CMOS, alimentazione 5V, driver di uscita in grado di erogare/assorbire 20 mA per ogni pin, diodi di camping verso alimentazione sugli ingressi, CPU risc a 8 bit, 35 istruzioni, 20 MHz di clock esterno massimo, 5 MIPs, 2k word di memoria FLASH programma, 224 byte di memoria RAM, 128 byte di memoria EEPROM dati programmabile da firmware, stack hardware a 8 livelli, comparatori analogici, con 16 step di tensione di riferimento con partitore tensione di alimentazione, 2 contatori a 8 bit, 1 contatore a 16 bit, un modulo comparatore/generatore di PWM, pull-up interni programmabili sugli ingressi, modalità risparmio energetico con arresto clock, 8 sorgenti di interrupt, USART[11] programmabile.

---

[10] *Cut & wire* indica le operazioni di modifica della topologia su un circuito già riportato in basetta, per piccoli errori di progetto o modifiche di cui ci rende conto solo in fase di prototipizzazione. *Cut* indica il taglio di una pista tracciata non più necessaria, *wire* l'inserimento di un filo a ponticello per realizzare un nuovo collegamento non previsto inizialmente.

[11] Forse si è più familiari con il termine UART: Universal Asyncronous Receiver Transmitter. La S sta per Syncronous, poiché la USART del PIC è in grado di realizzare trasmissioni sia

Architettura interna del PIC16F628

La potenza di elaborazione e la capacità di memoria del microcontrollore scelto sono le minime possibili per la complessità del problema ad esso demandato: questa scelta è stata fatta per raccogliere la sfida di produrre codice efficiente.

Architettura Harvard

La CPU del PIC legge le istruzioni da eseguire da una memoria FLASH, alla quale è collegata mediante un bus indirizzi e un bus dati, mentre l’accesso alla RAM (che nel gergo della Microchip viene detta *file registers*) avviene su una coppia bus dati/bus indirizzi separata. Questa architettura viene detta di Harvard, a differenza della tradizionale architettura di Von Neumann, in cui c’è un'unica coppia di bus dati/bus indirizzi dal quale vengono lette tanto le istruzioni che i dati. L’architettura Harvard è tipica dei microcontrollori in cui la memoria programma è a sola lettura, e presenta l’ulteriore vantaggio di poter avere larghezze di bus differenti per la memoria dati e la memoria programma, consentendo di avere gli opcode, coi loro operandi, tutti a singola word, anziché su più byte consecutivi, velocizzando le

---

sincrone (linee data + clock) che asincrone (un unica linea) liberando le risorse della CPU per

operazioni di fetch e garantendo tempi di esecuzione delle istruzioni sempre uguali. Nei PIC tutte le istruzioni vengono eseguite in 4 cicli di clock esterno, a parte le istruzioni di salto che vengono eseguite in 8 cicli.

Programmatore

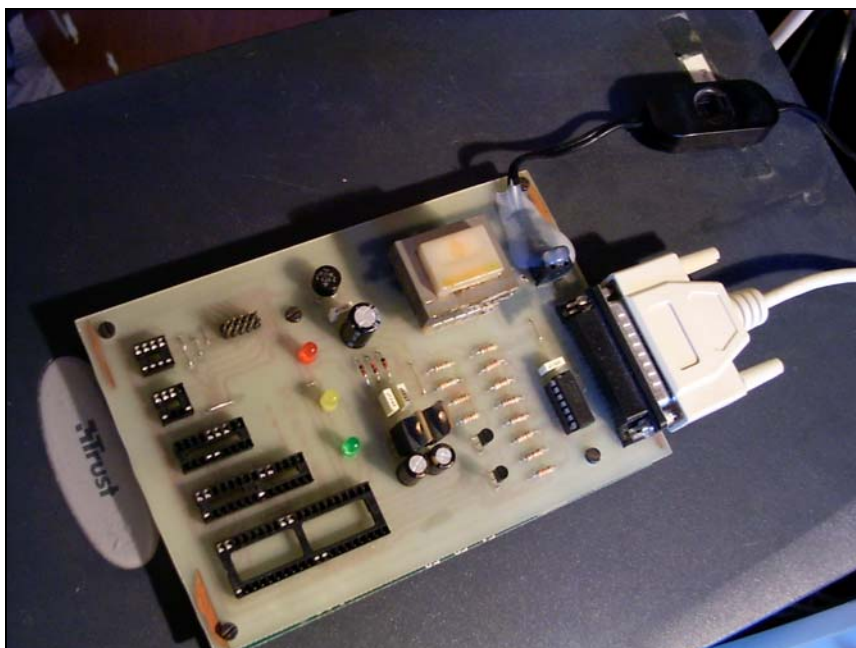
La memoria FLASH di programma del microcontrollore viene cancellata e riscritta dai piedini del chip mediante una interfaccia da collegare al PC, che prende il nome di programmatore. Esistono prodotti commerciali in grado di realizzare questa funzione, ma sono costosi, a volte hanno problemi di funzionamento, specie quelli economici, e spesso richiedono la rimozione del chip dal circuito per la riprogrammazione e il successivo reinserimento per testare se il firmware scritto funziona a dovere. I PIC consentono invece la possibilità dell'*in-circuit serial programming* (ICSP nel seguito), ovvero di programmare il microcontrollore direttamente nel circuito finale, ed è una possibilità fondamentale durante la fase di messa a punto del firmware.

Per risparmiare denaro, per poter contare su una assistenza clienti direttamente dal progettista, e per avere un prodotto funzionante meglio di tanti commerciali, si è deciso di autocostruire il programmatore seguendo un progetto disponibile in rete all'indirizzo http://www.jofi.it/fiser/ (o cercando Fiser's programmer su http://www.google.it/).

Il progetto è ben fatto e ben documentato, viene alimentato direttamente con la tensione di rete, e vanta (a conoscenza dell'autore) più di 400 esemplari costruiti nel mondo. E' fornito anche il disegno delle piste pronto da incidere su una basetta monofaccia. La basetta è stata realizzata col metodo *stira e ammira*[12], descritto più avanti, dopo aver ingrossato un po' le piste del disegno originale di Fiser[13] per adattarle alle tolleranze del mio metodo di incisione e dopo aver spostato appena i fori per il trasformatore, che aveva un rapporto di forma differente rispetto a quello utilizzato da Fiser. L'autore è stato ben lieto di fornire assistenza gratuita e spendere qualche tempo in chiacchiere sulle scelte progettuali per email: un netto vantaggio rispetto agli apparecchi commerciali.

altri scopi mentre la trasmissione è in atto.

[12] La dizione è in voga su news://it.hobby.elettronica, newsgroup italiano di riferimento per chi si diletta di elettronica applicata, e pare sia stata ideata da Celsius, un assiduo frequentatore del newsgroup.

[13] Fiser è un *nickname*, ovvero lo pseudonimo con cui l'autore è conosciuto in Internet.

Il mio esemplare autocostruito del Fiser’s programmer

Il progetto presenta diversi accorgimenti davvero geniali, non riscontrati nei prodotti commerciali di prezzo medio:

- sono previsti 5 fori sulla basetta per fissarla, con distanziatori esagonali, ad un’altra che ha l’unico scopo di proteggere e isolare dalla superficie di appoggio il lato piste della prima;
- l’alimentazione è direttamente a tensione di rete, e sulla basetta sono previsti fori per fissare saldamente il cordone di alimentazione per evitare che venga strattonato; tutti i prodotti commerciali richiedono invece un alimentatore DC apposito;
- non utilizza il costoso zoccolo textool (che si rompe dopo circa 200 azionamenti, poiché si consumano per sfregamento i listelli metallici di contatto), proponendo invece dei comunissimi zoccolini economici dai quali si sia avuta cura di rimuovere i pin non utilizzati durante la programmazione: i chip possono essere infilati e sfilati a mano, grazie a questo accorgimento, senza timore di romperne i piedini per torsione;
- il circuito è integralmente documentato e, se si rompe un componente, è facile trovare quello guasto e sostituirlo;
- sono previsti 3 led di segnalazione che monitorano la presenza dell’alimentazione, la presenza della tensione di 13,8V sul piedino di programmazione[14] e una delle due linee di scambio dati;

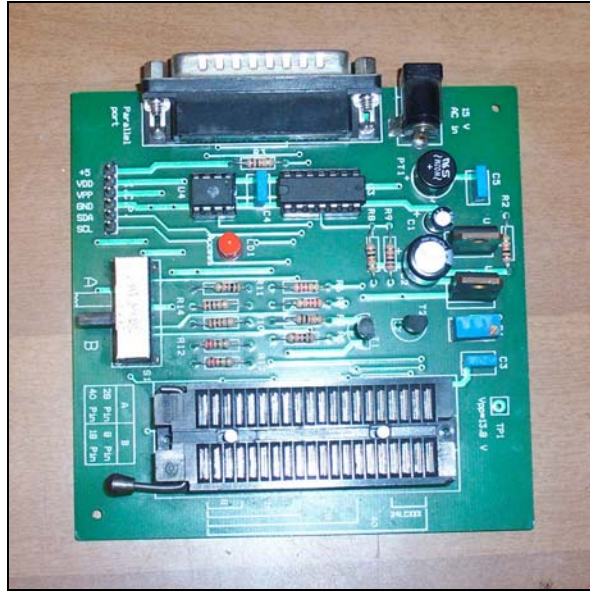
[14] I PIC funzionano normalmente in logica TTL 0/5V. Su un piedino è consentita la tensione di 13,8V che li porta in modalità programmazione, inibendo il funzionamento abituale.

- è disponibile l'header per ICSP che, oltre ai 5 canonici segnali, presenta altri 5 pin con il segnale di massa. Utilizzando un connettore per cavo piatto a 10 fili, si ottiene che ogni segnale è racchiuso tra due cavi posti a potenziale di massa da un lato e dall'altro, così che i segnali non si disturbino tra loro per l'effetto schermante ottenuto: è lo stesso accorgimento usato nel cavi IDE a 80 fili.

Per la descrizione dettagliata di questo circuito, il cui progetto non è mio, rimando al sito dell'autore, peraltro ben fatto, per dedicare più spazio ad altro in questa relazione.

Il programmatore è stato costruito e testato con file HEX[15] di esempio disponibili in rete e con il sofware EpicWin della microEngineering Labs. La versione beta del software è scaricabile dal sito http://www.melabs.com/ e utilizzabile liberamente per uso non commerciale. La licenza ha comunque un costo contenuto e viene spesso fornita in *bundle* con programmatori commerciali. In effetti io posseggo la licenza del programma proprio perché avevo acquistato il programmatore commerciale della Futura Elettronica (http://www.futuranet.it/).

Tale programmatore, oltre a non possedere nessuna delle caratteristiche elencate sopra, aveva un errore sul circuito che non gli consentiva di programmare tutti i PIC e che ho dovuto correggere per programmare i miei. Inoltre dopo un certo numero di utilizzi ha cessato di funzionare inspiegabilmente, e non sono riuscito ad avere assistenza dalla ditta per la riparazione, né a trovare nessun componente guasto a parte il microcontrollore pre-programmato presente sulla basetta. Per consolarmi ho cannibalizzato la basetta recuperandone i componenti.

[15] HEX è il formato dei file binari da inserire nella memoria FLASH dei microcontrollori; è un formato basato su ASCII con ogni riga contenterete indirizzo di memoria e dati da inserire nelle celle adiacenti in esadecimale. Il prodotto della compilazione di firmware per PIC è un file HEX.

Il programmatore commerciale acquistato dalla Futura,
“morto” dopo pochi mesi di utilizzo e non più riparabile.

MPASM Assembler

Impariamo a programmare i PIC

Una volta verificato che i microcontrollori venivano programmati ed era possibile rileggere il codice inserito dentro, a conferma che il programmatore funzionava bene, non restava che far pratica con il linguaggio di programmazione per poter inserire nei chip programmi sviluppati in proprio.

Ambiente di sviluppo integrato (IDE)

La Microchip stessa fornisce gratuitamente un compilatore assembler, chiamato MPASM, oggi disponibile nella versione 3, e un ambiente di sviluppo integrato, chiamato MPLAB, di cui ora è disponibile la versione 6 per Windows a 32 bit. Io ho utilizzato la versione 5.5 per Windows a 16 bit perché ho avuto problemi con nomi di path più lunghi di 64 caratteri con la versione 6. Nell’ambiente di sviluppo, oltre alla gestione di progetti[16], editor di testo a pieno schermo, compilatore integrato, è disponibile un simulatore e debugger in grado di simulare l’esecuzione del codice istruzione per istruzione e la stimolazione dei pin esterni del chip direttamente sul PC, e direttamente dall’ambiente di sviluppo integrato, prima di trasferire il codice sul microcontrollore reale.

[16] I progetti, negli ambienti di sviluppo, consentono di mantenere ordinati in un corpo unico diversi file correlati, ad esempio per una realizzazione complessa che prevede più microcontrollori programmati con codice differente.

Linguaggi di alto livello

Esistono dei tentativi, ora giunti a maturità, di linguaggi di programmazione più evoluti per i microcontrollori della Microchip. Certamente si tratta di un grosso aiuto alla stesura del codice, poiché si può utilizzare la sintassi del C o del Basic, con cui si ha più familiarità, anziché i criptici mnemonici assembler. Non ci sono solo vantaggi, però, poiché i costrutti più semplici sono disponibili anche in assembler, e i costrutti più avanzati, ovvero il controllo di flusso (strutture if-then, while, for, chiamate a sottoroutine con parametri), le operazioni aritmetiche con parentesi, la gestione delle stringhe, si pagano al caro prezzo di una occupazione spropositata di memoria programma con le librerie del linguaggio, e la perdita della possibilità di valutare il tempo di esecuzione delle istruzioni, spesso prerogativa fondamentale nei contesti real-time in cui vengono utilizzati i microcontrollori. Inoltre tutti i compilatori C, compreso quello della Microchip e l'unico compilatore Basic disponibile, non sono gratuiti[17], e non era disponibile il budget per l'acquisto di una licenza per questo progetto.

Assembler

Io ho scelto di utilizzare l'assembler[18], poiché consente un controllo più diretto della macchina e obbliga ad una conoscenza più dettagliata dell'hardware che consente poi di sfruttarlo al meglio. Il paradigma di programmazione in assembler è completamente differente da quello dei linguaggi di alto livello. Lo stesso problema viene risolto con costrutti e sequenze spesso completamente differenti, e sempre e comunque più efficienti, sia per velocità di esecuzione, sia per occupazione di memoria (una risorsa molto preziosa sui microcontrollori) in assembler. Non è detto che il paradigma del linguaggio macchina sia più complesso di quello dei linguaggi di alto livello: è semplicemente differente: chi è abituato a programmare in un modo avrà difficoltà a programmare nell'altro, ma vale la pena di imparare entrambe le possibilità.

---

[17] Fa eccezione qualche compilatore C completamente freeware, ma si tratta di prodotti ancora in stato di beta testing e pieni di bug che ne rendono impossibile l'uso.

[18] Qualche purista potrebbe obiettare che assembler è il nome da dare al compilatore (chiamarlo compilatore, secondo i puristi, sarebbe troppo poiché effettua solo traduzione 1-a-1 da mnemonici a codice binario), assembly il nome del linguaggio. Oggigiorno nessuno più presta attenzione a questa distinzione, e io userò il termine assembler per il linguaggio e il termine compilatore per il compilatore/assemblatore.

Domare l’assembler

La mancanza di strutture per il controllo di flusso (if-then-else while-do repeat-until case-switch) è stata risolta con la stesura di alcune macro che realizzano tali funzioni, rendendo così il codice assembler, che per sua natura tende a diventare *spaghetti programming*[19], un po’ più leggibile e mantenibile. Alla mancanza della gestione di stringhe si ovvia con tavole di salto per le costanti, e routine ad hoc per le stringhe vere e proprie da modificare in run-time. Per la gestione degli array occorre far ricorso all’indirizzamento indiretto disponibile sul PIC. Non c’è una soluzione immediata per l’aritmetica complessa, ma esistono librerie pronte in rete per i problemi più comuni. Ciò di cui si sente davvero la mancanza in assembler sono i passaggi di parametri di ingresso e uscita a sottoprocedure, poiché la cura dell’impegno di ciascuna locazione di memoria per ogni istante di esecuzione (ovvero l’attenzione alla durata delle variabili) in assembler è affidata interamente al programmatore. Nei miei programmi dedico uno spazio di memoria ad ogni sottoroutine, che usa le sue variabili, così è certo che non si verifichino sovrapposizioni. Si spreca così un po’ di memoria RAM poiché una locazione non più utilizzata da una routine poteva essere utilizzata da un’altra, ma si evitano sovrapposizioni. Per quanto riguarda il passaggio di parametri in ingresso, nessun problema si pone se occorre passare un unico valore a 8 bit: è disponibile l’accumulatore, o se le sotto-procedure sono minuscole: in tal caso le si scrive come macro[20], e queste si, possono accettare diversi parametri di ingresso.

Delle altre funzioni dei linguaggi di alto livello, mere traduzioni di quanto l’hardware del PIC mette a disposizione attraverso i registri di controllo, francamente non se ne sente la mancanza, poiché basta nominare i registri con nomi opportuni o scrivere piccole routine dai nomi evocativi, per ottenere la stessa facilità d’uso dei linguaggi d’alto livello in assembler.

Riferimenti per l’assembler dei PIC

Ci vuole del tempo per imparare l’assembler dei PIC, che utilizza mnemonici differenti da quelli dei processori x86 o da altri processori storici (6502 o Z80), e non è questa relazione la sede adatta per descrivere tale linguaggio. Si rimanda al data-sheet del PIC16F628 della

---

[19] Uno *spaghetti program* è un codice in cui l’esecuzione salta continuamente da un punto ad un altro tramite salti incondizionati o condizionati: in un tale codice è difficile tenere traccia dei percorsi di esecuzione così come è difficile seguire con lo sguardo i ripiegamenti degli spaghetti in un piatto di pasta fumante.

[20] Una macro è un nome simbolico per un frammento di codice che verrà reinserito dal compilatore ogni volta che viene richiesto, occupando così tanta più memoria quante più volte viene richiamato. Si differenzia dalle sottoroutine che sono presenti in memoria una volta sola e sfruttano lo stack hardware per tornare all’uscita alla procedura chiamante corretta.

Microchip, fonte dalla quale ho imparato io, poiché avevo già una piccola esperienza in altri linguaggi macchina[21] e mi servivano solo informazioni specifiche su questa architettura, all'help in linea dell'assembler MPASM della Microchip, e al libro PicBook, disponibile gratuitamente in lingua inglese sul Web all'indirizzo http://www.mikroelektronika.co.yu/english/product/books/PICbook/ e a pagamento in lingua italiana in libreria (la versione italiana, inoltre, contiene molto meno materiale dell'originale in inglese).

Microcontrollore-resistenza-led [22]

Prima di affrontare il progetto completo si è testato il microcontrollore con piccoli circuiti che hanno consentito di prendere familiarità con le possibilità disponibili, operando in un contesto più controllato in cui era difficile rompere qualcosa (per esempio con le extra tensioni di apertura del carico induttivo della bobina dei relais): led collegati ai piedini di uscita.

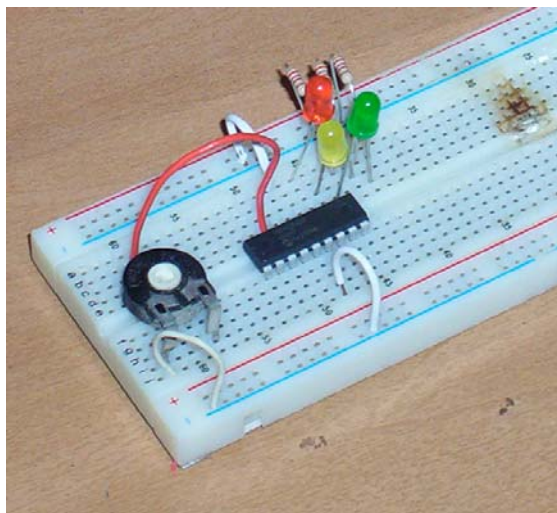
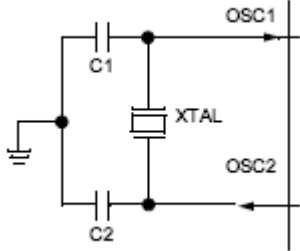
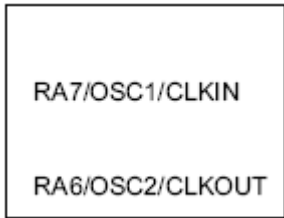
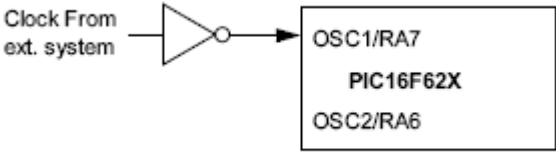
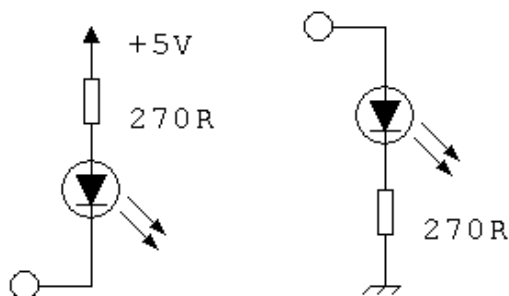
Le sperimentazioni, realizzate su *solderless board*[23], hanno esplorato le differenti sorgenti di clock disponibili per il microcontrollore (oscillatore interno, quarzo, rete RC con resistore esterno, condensatore interno, sorgente di clock esterna) e le due diverse modalità di utilizzo di una uscita del microcontrollore per generare o assorbire corrente, oltre a far prendere dimestichezza con il set di istruzioni del microcontrollore.

Per non appesantire la trattazione, evito di riportare il codice assembler e lo schema elettrico di questi primi esperimenti.

---

[21] Assembly del 6502 utilizzato sul Commodore 128 in tenera età, per velocizzare l'esecuzione di propri programmi e videogiochi e Assembly del 80286 utilizzato come codice in-line in programmi Pascal per l'accesso diretto alle funzioni di basso livello di BIOS e DOS.

[22] Il titolo di questo paragrafo è la parafrasi del più celebre "pila-resistenza-led", ovvero il primo circuito che l'hobbista di elettronica applicata realizza nella sua carriera, e che lo porta per la prima volta dal mondo dell'elettrotecnica (pila-interruttore-lampadina) nel mondo dei semiconduttori (il led).

[23] Basette sperimentali con fori distanziati 1/10 di pollice (il passo dei piedini dei circuiti integrati in package PDIP). In ciascun foro è presente una pinzetta argentata che afferra i reofori dei componenti. Le pinzette sono già collegate internamente a strisce di 5, così che sia possibile montare un intero circuito senza effettuare saldature.

Uno dei circuiti più semplici realizzabili: sequenza di accensione delle luci di un semaforo per plastico ferroviario, con potenziometro per regolare la velocità di scansione della sequenza.

I due tipi di collegamento di un led ad uno stadio di uscita CMOS push/pull.
In caso di uscita a drain aperto, è possibile utilizzare solo il collegamento di sinistra.

Un clock per il PIC

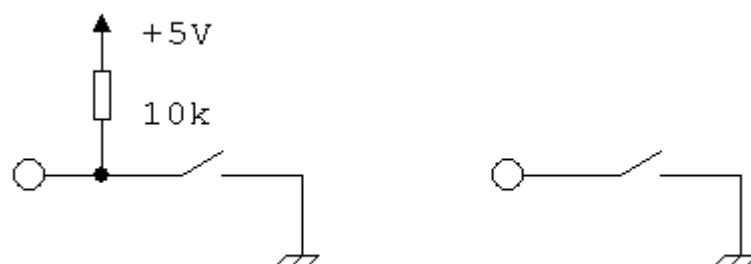
Ogni microcontrollore ha bisogno di una sorgente di clock per scandire il ritmo di esecuzione delle istruzioni. I PIC offrono grande versatilità consentendo di selezionare, in fase di programmazione, tra numerose sorgenti di clock disponibili, e includono già la circuteria di oscillazione. Qualunque sia la sorgente di clock, il tempo di esecuzione di una istruzione è 4 colpi di clock per tutte le istruzioni tranne quelle di salto (goto return call retfie) che impiegano 8 colpi di clock per essere eseguite. Questa uniformità di tempo di esecuzione, che rende facile calcolare il tempo di esecuzione delle routine contando le istruzioni presenti, è possibile grazie all'architettura Harvard e al set di istruzioni RISC.

	Quarzo esterno. E’ indispensabile quando la frequenza di esecuzione delle istruzioni deve essere precisa, come nel nostro caso, poiché si intende usare la comunicazione seriale.
	Rete RC con resistore esterno. Oltre al vantaggio del costo contenuto, si può lavorare a frequenza regolabile sostituendo il resistore con un semplice potenziometro.
	Oscillatore interno a frequenza fissa: i due pin del PIC possono essere utilizzati per altre funzioni. Utile quando è necessario disporre di molti piedini di ingresso/uscita.
	Sorgente di clock esterna. Utilizzabile quando è già presente un clock sul circuito.

Alcune delle differenti sorgenti di clock disponibili per il PIC16F628

La sorgente di clock si imposta durante la programmazione attraverso la configuration word, una particolare parola di memoria FLASH in cui si inseriscono dati di configurazione per i circuiti integrati (che io chiamerò nel seguito “gadget hardware”) presenti nel microcontrollore.

Resistenze di pull-up integrate

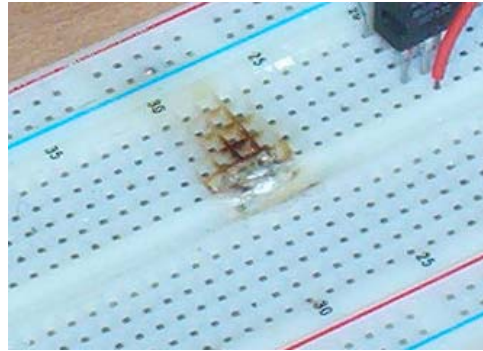
In una sperimentazione successiva sono stati testati gli stadi di ingresso con semplici interruttori, apprezzando la funzionalità dei pull-up interni che consente di risparmiare una resistenza sulla basetta.

Resistenze di pull-up esterne (a sinistra) ed interne (a destra)

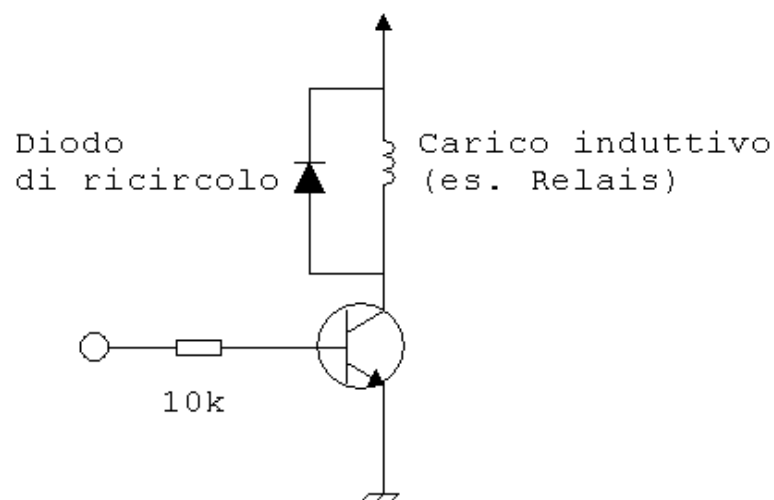
Normalmente, per generare i due stati logici 0V e 5V per un circuito logico con un interruttore, si utilizza il collegamento di sinistra, con una resistenza di pull-up verso l'alimentazione positiva e l'interruttore che collega l'ingresso a massa. Alcuni ingressi del PIC hanno però pull-up interni, inseribili da programma: è così sufficiente collegare un interruttore al piedino di ingresso e a potenziale 0V per poterne leggere lo stato, come nel circuito di destra.

La soddisfazione ottenuta con i circuiti semplici sprona ad affrontare con più serenità e con più cognizione di causa progetti più impegnativi. Procedendo per gradi, si accumulano soddisfazioni su soddisfazioni. Affrontando direttamente progetti complessi, invece, si rischia di dover rifare il lavoro due volte e si accumulano fallimenti e delusioni.

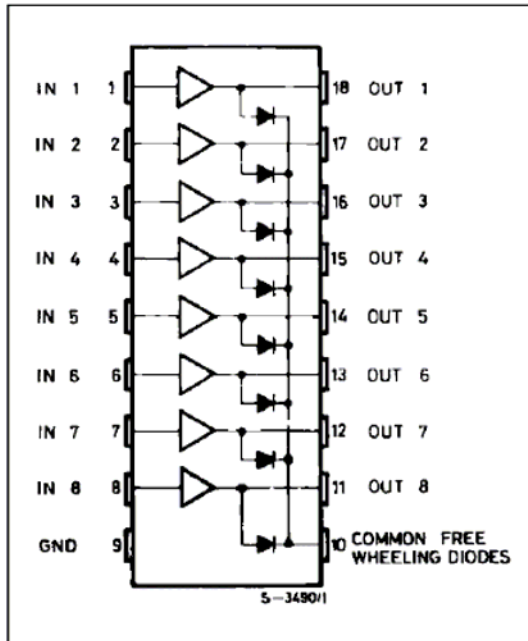
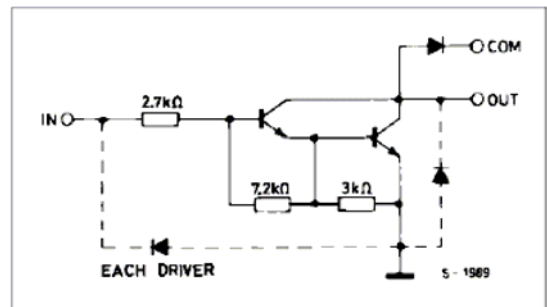
Pilotare relais con un microcontrollore

La corrente erogata da uno stadio di uscita di un PIC non è in grado di eccitare la bobina di un relais (circa 80-100 mA per i relais più comuni). Anche se esistono relais miniatura con correnti di 15mA, è comunque sconsigliabile collegarli direttamente alle uscite del PIC, poiché le extra-tensioni di apertura[24] possono superare diverse migliaia di Volt, distruggendo gli stadi di uscita del chip.

[24] Una bobina in cui scorre corrente genera un campo magnetico. Togliendo l'alimentazione, il campo magnetico induce una corrente che, se non si chiude su nessun carico, porta ad un incremento della differenza di potenziale ai capi della bobina.

Alcuni dei fori della breadboard, sciolti dalla fusione
di un transistor che eccitava un relais senza protezioni

E necessario quindi inserire un transistor con la base opportunamente polarizzata da una resistenza per prelevare poca corrente dal microcontrollore nel circuito. E' anche una buona idea inserire un diodo di ricircolo che fornisca un percorso alternativo alla corrente di apertura della bobina, così che questa non rompa il transistor ma fluisca nel diodo. Il diodo deve essere veloce, cioè commutare dall'interdizione alla conduzione più rapidamente del transistor, e quindi deve essere stato costruito con una tecnologia differente rispetto al transistor.

Metodo corretto per collegare un relais allo stadio di uscita push-pull di un chip CMOS.

La resistenza si dimensiona decidendo quanta corrente far erogare al PIC, affinché il transistor lavori in zona di saturazione con la corrente di collettore necessaria a tenere acceso il relais. Per un relais da solo va bene utilizzare componenti discreti. Se il numero di relais aumenta, è

una buona idea utilizzare il chip ULN2803, che contiene all’interno 8 transistor Darlington[25], le relative resistenze di polarizzazione per l’uso con circuiti logici CMOS alimentati a 5V, e i diodi di *clamping*[26] per ogni uscita.

La piedinatura dell’ULN2803 e lo schema elettrico di ciascuno degli 8 driver

Shift register con latch

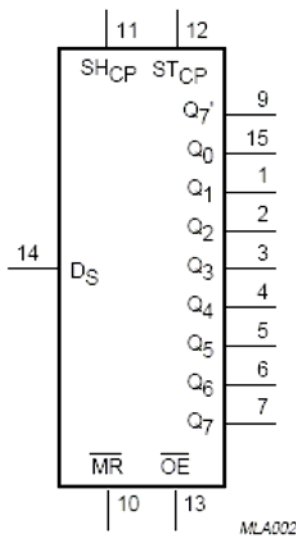
Per aumentare il numero di uscite digitali presenti su un microcontrollore, la scelta più ovvia è utilizzare degli shift-register. Io ho optato per i popolari **74HC595**, shift-register ad 8 bit in tecnologia CMOS High-Speed. Questi sono in grado di lavorare a frequenze di clock dell’ordine dei 100 MHz, hanno gli stati di uscita tri-state[27], un piedino con la replica dell’ultima uscita non bufferata dai latch per consentire il collegamento di più shift-register in cascata, i bit di uscita disponibili su pin tutti dallo stesso lato del componente e in ordine su pin adiacenti[28] (non è una caratteristica di secondaria importanza se si intende fare uno

[25] La topologia Darlington connette due transistor in modo da fornire guadagni di corrente elevatissimi (10^3, 10^4). Nel nostro caso, verrà assorbita pochissima corrente dallo stadio di uscita del circuito logico.

[26] Si usa il termine *clamping* per indicare un diodo collegato, con funzioni di protezione, verso le alimentazioni. Con questa topologia il diodo di clamping si comporta anche da diodo di ricircolo.

[27] Ovvero possono essere posti in modalità ad alta impedenza.

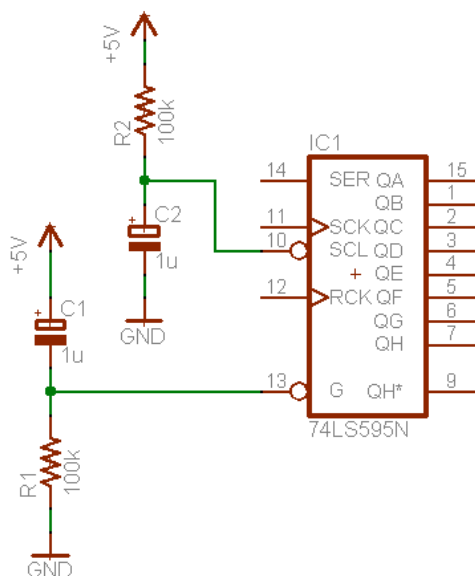
[28] Questa caratteristica non è onorata in tutti i circuiti integrati. Ad esempio nel 4017 (contatore decimale di Johnson) le 10 uscite sono “sparpagliate” tra i pin ed è difficile ottenere uno sbroglio ordinato.

sbroglio monofaccia), sono dotati di latch, ovvero ritengono una configurazione sugli stadi di uscita mentre lo shift-register ne sta leggendo un'altra serialmente.

Piedinatura del 74hc595

Le linee di ingresso più importanti sono D_S (Data serial), su cui inviare serialmente, un bit per volta, i dati da memorizzare, e SH_{CP} (Shift Clock Pulse) segnale di clock per gli shift-register: in corrispondenza del fronte di salita di questo clock i dati verranno trasferiti da ogni bit al successivo e dall'ingresso Data al primo bit. Un fronte di salita sul segnale ST_{CP} (Store Clock Pulse) memorizzerà invece la configurazione attuale nei latch, dai quali verrà proposta sui pin di uscita attraverso stadi push-pull CMOS 3-state. OE/ (Output Enable Active Low), posto alto, rende tutti i pin di uscita ad alta impedenza, mentre va tenuto basso nel funzionamento normale. MR/ (Master Reset Active Low) azzera asincronamente gli shift register.

Anche questo componente è stato oggetto di sperimentazione su *solderless board* prima dell'uso nel progetto finale. Più che il componente, si sono testate le routine del microcontrollore adatte a pilotarlo correttamente. Data la velocità più elevata dello shift-register (100MHz = 10ns) rispetto alla velocità di esecuzione di una istruzione del microcontrollore (5 MHz = 200ns), non è stato necessario introdurre alcun ciclo di ritardo nel firmware del PIC. Il tempo necessario per aggiornare serialmente l'intera sequenza degli stati di 64 relais, è inferiore al tipico tempo di risposta di un relais (dell'ordine dei millisecondi), quindi questo tipo di problemi (cicli di ritardo per componenti lenti, aggiornamento eventualmente troppo lento degli shift-register rispetto al tempo di commutazione del relais) si è risolto da solo prima ancora di porsi, avendo a disposizione ampi margini di lavoro su tutti i fronti.

Problema dello stato iniziale

Poiché il PIC appena alimentato ha tutti i suoi piedini configurati come ingressi (successivamente il firmware ne configura alcuni come uscite), e quindi si trovano in alta impedenza, e poiché anche gli ingressi del 74HC595 sono ad alta impedenza, le relative linee, all'accensione del circuito, sono flottanti, e può capitare che vengano letti stati logici 0 e 1 alternativamente e che lo shift-register si riempia di dati casuali. Se gli shift register sono collegati a carichi, questi potrebbero inserirsi non volutamente, durante la fase di accensione del circuito.

Reti RC sui piedini di reset e abilitazione uscite

Volendo garantire che all'accensione del circuito gli shift register siano tutti a zero e gli output non siano attivati prima che il resto dell'elettronica non sia alimentata correttamente e prenda il controllo della situazione, si può prevedere un circuito con un condensatore e una resistenza sugli ingressi Master Clear e Output Enable. Appena acceso il circuito, il gradino di tensione sull'alimentazione si trasferirà dall'altro lato del condensatore sul piedino di ingresso dello shift register, poi la resistenza scaricherà o caricherà il condensatore a massa o alimentazione, mantenendo il valore finale per tutto il resto del tempo in cui il circuito rimane acceso.

Si ottiene così un 1 o uno 0 all'accensione, a seconda di quale delle due reti mostrate in figura è utilizzata, e lo stato opposto per tutto il resto del tempo. Resistenza e condensatore possono essere dimensionati per ottenere i tempi di reazione desiderati. Questo non è un problema per il nostro circuito perché il microcontrollore prende il controllo del bus molto rapidamente e per prima cosa invia una sequenza di zeri agli shift-register. I relais, che essendo componenti elettromeccanici sono tre ordini di grandezza più lenti degli shift-register e del microcontrollore, non hanno il tempo di scattare, quindi si è deciso di semplificare il circuito collegando permanentemente ad alimentazione e massa gli ingressi Master Clear e Output Enable[29], fidandosi del fatto che il microcontrollore prende rapidamente il controllo del bus, ottenendo il vantaggio di una semplificazione del circuito.

[29] Nell'ultimo schema, relativo allo stesso componente, la nomenclatura degli ingressi è differente: SER (Serial) al posto di D_S (Data serial), SCK (Serial Clock) al posto di SH_{CP} (Shift Clock Pulse), RCK (Retain Clock) al posto di ST_{CP} (Store Clock Pulse), SCL (System Clear) al posto di MR/ (Master Reset Active Low), G (Gate) al posto di OE/ (Output Enable Active Low). Le uscite sono indicate con A B C D E F G H anziché con 0 1 2 3 4 5 6 7. Inoltre sono evidenziati i clock con il triangolino e le uscite a logica invertita con il pallino, anziché con la sbarra o la sovralineatura. Questo non deve stupire poiché in elettronica

Comunicazione seriale

La seriale RS-232 è una porta *legacy*, termine inglese con il quale si indicano gli standard veri e propri o standard *de facto* così consolidati negli anni da essere presenti sulla quasi totalità dei sistemi elettronici e per i quali ci si aspetta un supporto pieno senza particolari complicazioni o intoppi poiché, esistendo da molti anni, sono ben sperimentati e sono poco esigenti per l'hardware più recente, che può implementarli senza costi aggiuntivi particolarmente gravosi.

Quasi tutti i PIC della microchip integrano una UART, il che significa che la CPU deve solo configurare la UART, e passarle i dati da trasmettere, poi può occuparsi di altro mentre la trasmissione è in corso. Similmente, in ricezione, la UART accetterà i dati di arrivo e li manterrà in un buffer fino alla richiesta di lettura della CPU, che nel frattempo può essersi occupata di altro. La presenza di una UART hardware è un grosso vantaggio per il programmatore che deve preoccuparsi unicamente della configurazione e della gestione ad alto livello della comunicazione (per esempio il controllo di flusso) senza preoccuparsi delle questioni inerenti il mezzo fisico (campionamenti multipli per interpretazione a maggioranza di dati disturbati, sincronizzazione con l'inizio del messaggio e simili).

Per una descrizione del pattern NRTZ (Non Return To Zero) tipico della trasmissione seriale, della sincronizzazione byte per byte grazie ai bit di start e di stop, dei campionamenti multipli su ciascun bit, per decisioni a maggioranza in caso di campionamento di valori diversi sullo stesso time-slice dedicato ad un bit, dell'architettura della porta UART del PIC, si veda il data sheet del PIC16F628, e la numerosa letteratura sull'argomento presente su Internet, poiché non è questa relazione la sede adatta ad approfondire queste questioni.

Livelli di tensione della porta seriale

Lo standard *de facto* RS-232 prevede tensioni positive e negative rispetto a massa per indicare i due stati logici. Lo standard indica che una tensione inferiore a -3V debba essere prodotta per indicare un 1, e una superiore a +3V debba essere prodotta per indicare uno 0. In ricezione le soglie sono ridotte a -0,3V e +0,3V, poiché il segnale può attenuarsi durante il tragitto. Poiché lo standard prevede una tensione minima, le porte seriali di solito utilizzano tensioni maggiori di 3V per garantire una maggiore immunità ai disturbi e consentire la trasmissione dati a distanze superiori. Tipicamente le tensioni in uso sono comprese tra 8V e 12V sui PC da tavolo, 5V sui portatili, per risparmiare sulle batterie.

---

esistono diversi standard per il disegno degli schemi elettrici, ed è bene, didatticamente, prendere confidenza con tutti.

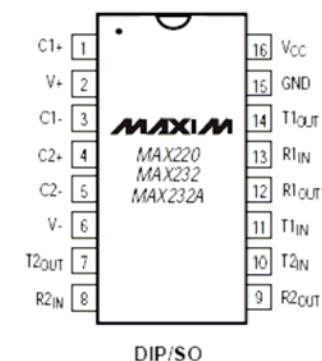
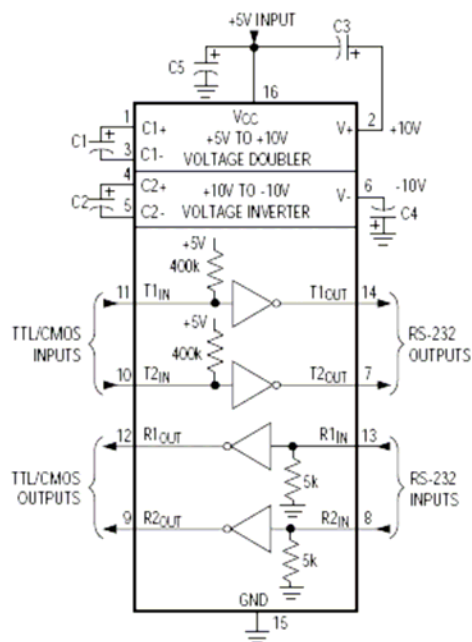
Il PIC lavora in logica TTL a 5V, ovvero usa 0V per indicare lo zero logico e 5V per indicare l'1 logico. Esistono tecniche per collegare direttamente il PIC alla porta seriale senza adattatori di livello, che fanno affidamento sul fatto che le porte seriali dei PC di solito leggono correttamente come tensione negativa (un 1 logico) anche tensioni di 0V o debolmente positive. Queste tecniche sono di tipo *bit-banging*[30], ovvero non si appoggiano sulla UART hardware del PIC, poiché devono invertire la logica di funzionamento, ma generano e interpretano i bit leggendoli direttamente dalle uscite e ingressi logici per uso generico, con del codice scritto appositamente.

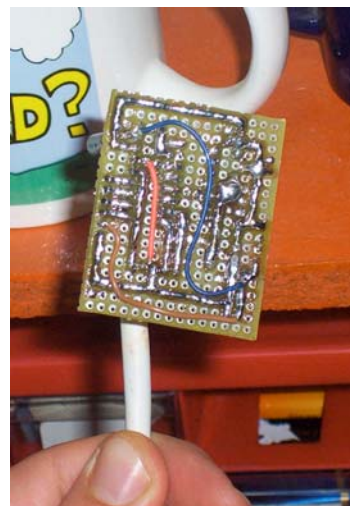
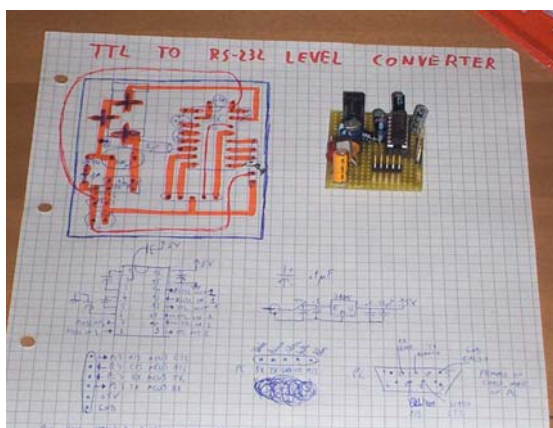
Queste tecniche sono interessanti poiché consentono di risparmiare componenti, ma non raggiungono l'immunità al rumore e la precisione delle temporizzazioni di una UART hardware. Inoltre sono fuori standard poiché non tutte le porte seriali interpretano come negativa una tensione di 0V, e quindi non funzionano in tutti i contesti operativi.

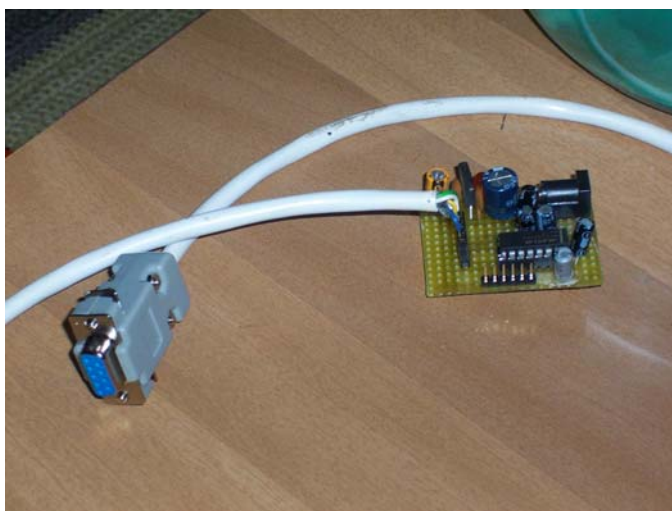
Il traslatore di livello Max232

Per il nostro circuito, si è deciso di utilizzare la UART del PIC con un adattatore di livello esterno che trasformasse i livelli 0V/5V tipici del PIC in livelli +10V/-10V della porta seriale. La Maxim (http://www.maxim-ic.com/) ha una serie di adattatori di livello integrati già pronti per diversi usi. E' stato scelto il MAX232 (un nome, un programma), che integra oltre ai traslatori di livello, anche un duplicatore di tensione per portare i 5V di alimentazione a 10V, e un invertitore di tensione per portare i 10V a -10V. Per far funzionare questi due circuiti interni gli unici componenti esterni da montare sono condensatori elettrolitici, che non possono essere integrati. Il MAX232 è in grado di traslare di livello 2 ingressi e 2 uscite.

Sul data sheet è riportato lo schema elettrico tipico di collegamento.

[30] Trad. letterale: "spara-bit", nel senso che i bit vengono messi sulla linea direttamente dal Firmware, anziché da un circuito hardware dedicato.

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Pinout e schema elettrico applicativo tipico del MAX232

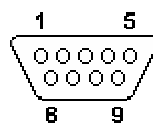
Sebbene sia possibile montare un max232 con gli elettrolitici esterni di contorno su una *solderless board*, si è preferito montare il circuito su una millefori, così che fosse possibile utilizzarlo facilmente sui propri circuiti sperimentali che avessero bisogno della porta seriale senza doverlo rimontare ogni volta. Data la facilità con cui è possibile inviare caratteri sulla seriale tramite il PIC, è infatti possibile utilizzare proficuamente questa possibilità durante il debug del firmware, lasciando che il programma in test avvisi dei punti del codice eseguiti man mano inviando caratteri di segnalazione. Il circuito costruito su millefori può essere connesso al volo al piedino di trasmissione seriale del PIC da un lato e alla porta seriale del PC su cui gira il terminale di Windows dall’altro, per il debug del firmware tramite la seriale in circuiti che non la utilizzeranno nella versione definitiva.

Schizzo su carta e montaggio su millefori

Il circuito è stato rapidamente schizzato su carta quadrettata, utilizzando proficuamente un evidenziatore per indicare le saldature, e montato su una millefori, cercando di realizzare con piste di stagno la maggioranza dei collegamenti, in modo da lasciare pochi a fili volanti. Sul circuito è presente il regolatore di tensione 7805 e un ponte a diodi, così che lo si possa alimentare indifferentemente con tensioni da 8V a 18V in DC, senza curarsi della polarità, o da 5V a 15V in AC. Un cavo seriale si collega tramite un connettore ad innesto rapido al circuito, mentre un header a 5 pin rende disponibili le linee di trasmissione e ricezione nonché due linee per il controllo di flusso hardware e il riferimento di massa. Per essere un circuito finalizzato alla sperimentazione col MAX232 e il test rapido del funzionamento della UART di un PIC, è fin troppo complesso!

Il circuito adattatore di livello consente di testare rapidamente la porta UART di un PIC.

La sperimentazione con la UART del PIC e un PC su cui girava il terminale di Windows, ha mostrato che velocità di 57600 baud[31] sono tranquillamente sostenibili anche da un comune cavo schermato a due poli lungo 40 metri. Velocità di 115200 baud o superiori invece vengono raggiunte con difficoltà su cavi di lunghezza elevata o di scarsa qualità. Si è così deciso di utilizzare la velocità di 57600 come buon compromesso tra rapidità di trasmissione e affidabilità anche su distanze più elevate.

Cavo due poli + schermo, lungo 40 metri, con connettori DSUB-9, utilizzato negli esperimenti per determinare la distanza massima raggiungibile affidabilmente per ogni velocità di trasmissione seriale

Il nostro circuito monterà un connettore a vaschetta a 9 pin DSUB-9 maschio, poiché quelli a 25 pin sono in disuso da tempo. Esistono comunque degli adattatori per adattare un cavo di un tipo ad un connettore dell'altro tipo.

I segnali presenti sul connettore a 9 pin sono i seguenti:

connettore DSUB9 maschio visto da fuori[32]

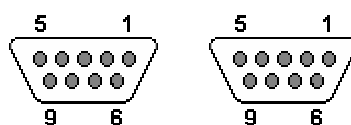
[31] Baud indica il numero di bit fisici (transizioni di livello) trasmessi per secondo, bps (bit per secondo) il numero di bit di dati trasmessi per secondo.

[32] I diagrammi delle porte seriali e le tabelle della descrizione dei pin, e del cavo di collegamento null-modem, sono state prelevate, per gentile concessione, dal sito http://www.hardwarebook.net/.

Pin	Name	Dir	Description
1	CD	←	Carrier Detect
2	RXD	←	Receive Data
3	TXD	→	Transmit Data
4	DTR	→	Data Terminal Ready
5	GND	—	System Ground
6	DSR	←	Data Set Ready
7	RTS	→	Request to Send
8	CTS	←	Clear to Send
9	RI	←	Ring Indicator

Segnali con relative direzioni, presenti sulla porta seriale

Nella nostra applicazione utilizzeremo solo i segnali RXD TXD (che chiameremo semplicemente RX e TX), il riferimento di massa sul pin 5, ed eventualmente i due segnali per il controllo di flusso hardware RTS, CTS, mentre non useremo gli altri segnali (DTR, DSR, RI, CD).

Il cavo di collegamento dovrà collegare le uscita della porta del PC con gli ingressi della porta sul circuito e viceversa, si potrà pertanto utilizzare un cavo null-modem[33], con il seguente schema di collegamento[34]

Connettori D-Sub9 femmina A e B, alle due estremità del cavo.

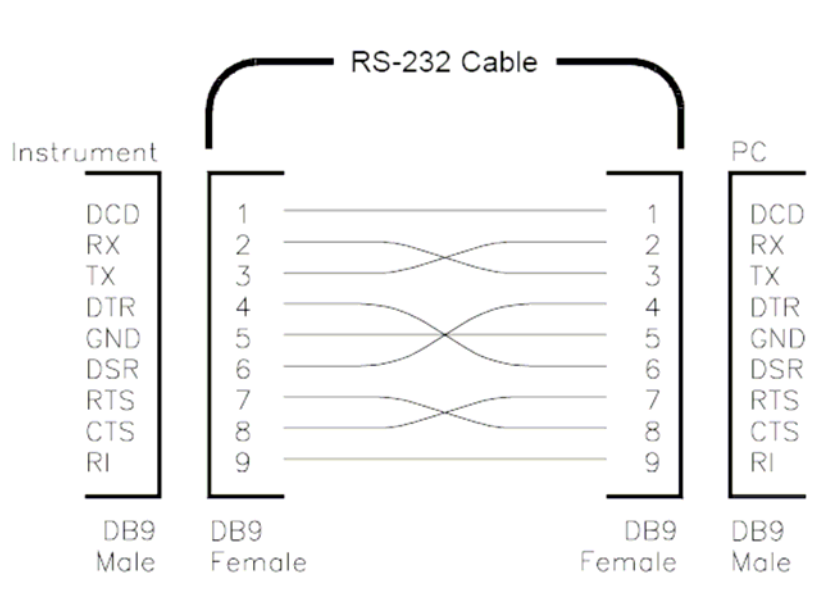
	D-Sub9 A	D-Sub9 B	
Receive Data	2	3	Transmit Data
Transmit Data	3	2	Receive Data
Data Terminal Ready	4	6+1	Data Set Ready + Carrier Detect
System Ground	5	5	System Ground
Data Set Ready + Carrier Detect[35]	6+1	4	Data Terminal Ready
Request to Send	7	8	Clear to Send
Clear to Send	8	7	Request to Send

Schema di collegamento classico di un cavo null-modem

[33] Detto anche *crossed*.

[34] Dato che i segnali più importanti, Trasmit e Receive sono localizzati sui pin 2 e 3, qualcuno sostiene che il numero d'ordine presente nel nome dello standard (RS232) serva proprio come mnemonico dei pin utilizzati per la trasmissione e ricezione.

[35] I segnali DSR e CD sono collegati assieme, sui cavi null-modem, per far credere a ciascuno dei due dispositivi di essere ondine: quando ci si collega ad un DCE (un modem), il segnale CD indica infatti l'avvenuta connessione; collegando tra loro due DTE (due PC), è bene che entrambi gli strumenti credano che l'altro sia sempre online.

Sul data-sheet della macchina Agilent di riferimento, consigliano un altro tipo di connessione, leggermente differente.

Schema di collegamento suggerito dall'Agilent

Nella nostra applicazione non utilizzeremo tutti i segnali, ed entrambi i tipi di cavo funzioneranno bene con il nostro apparecchio. Ci si potrebbe anche limitare ad un cavo schermato a 4 poli, secondo il seguente schema ridotto:

	D-Sub9 A	**D-Sub9 B**	
Receive Data	2	3	Transmit Data
Transmit Data	3	2	Receive Data
Data Terminal Ready Data Set Ready Carrier Detect	4+6+1: L/B	4+6+1: L/B	Data Terminal Ready Data Set Ready Carrier Detect
System Ground	5	5	System Ground
Request to Send	7	8	Clear to Send
Clear to Send	8	7	Request to Send

Covo ridotto: 4 poli + schermo

I pin 1, 4 e 6 vanno collegati tra loro in *loopback* su ciascun lato del cavo, ma non portati da un apparecchio all'altro. Volendo usare un cavo schermato con soli 2 poli, si possono collegare in *loopback* anche i pin 7 e 8: si rinuncia al controllo di flusso hardware, ma se si studia un protocollo adeguato (come in effetti è stato poi fatto) e si è certi che non si verifichino perdite di dati, questo non è un problema e la soluzione riduce la complessità e il costo del cavo, specie se va autocostruito.

	D-Sub9 A	**D-Sub9 B**	
Receive Data	2	3	Transmit Data
Transmit Data	3	2	Receive Data
Data Terminal Ready Data Set Ready Carrier Detect	4+6+1: L/B	4+6+1: L/B	Data Terminal Ready Data Set Ready Carrier Detect
System Ground	5	5	System Ground
Request to Send Clear to Send	7+8: L/B	7+8: L/B	Request to Send Clear to Send

Cavo minimo: 2 poli + schermo

I tipi di controllo di flusso standardizzati della porta seriale

Per controllo di flusso si intendono quelle tecniche atte a rallentare la trasmissione di dati di un dispositivo, quando il buffer di ricezione a destinazione, sull'altro dispositivo, è pieno o quasi pieno, in modo da prevenire la perdita di dati dovuta ad errori di buffer overflow, che nel caso specifico vengono di solito chiamati overrun. I controlli di flusso standardizzati per le porte seriali sono i seguenti:

XON/XOFF: E' un controllo di flusso software: due caratteri speciali sono inseriti nel flusso di dati per indicare la disponibilità a ricevere dati o la richiesta di fermare le trasmissioni in atto. Questi caratteri sono 0x13 (XOFF), che richiede di interrompere il flusso di dati, e 0x11 (XON) che è la richiesta a riprendere le trasmissioni. Il vantaggio di questo tipo di controllo di flusso è che sono sufficienti le due linee TX e RX e nessuna altra linea hardware per implementare il controllo di flusso. Lo svantaggio è che due caratteri del codice ASCII sono riservati al controllo del flusso e accorrono accorgimenti particolari se si intendono trasferire dati binari che possono comprendere tali caratteri (solitamente si raddoppia il carattere, se è presente nel *payload*).

RTS/CTS: E' un controllo di flusso hardware: ciascuna delle due parti setta la linea RTS quando ha spazio nel buffer di ricezione, e si astiene dall'inviare dati se la linea CTS non è bassa. Il vantaggio di questo tipo di controllo di flusso è che è di facile implementazione in sistemi *embedded* come quelli a microcontrollore, pur richiedendo due linee dedicate.

DTR/DSR: Funziona allo stesso modo di RTS/CTS, ma usa altre due linee disponibili sulla porta seriale. I **modem** usano tipicamente una combinazione dei due controlli di flusso disponibili, riferendo la coppia di segnali DTR/DSR all'avvio del software di controllo sul PC e del firmware di controllo sul modem, e la coppia RTS/CTS al controllo di flusso vero e proprio legato ai buffer.

Nessun controllo di flusso: per velocità di comunicazione non elevate, o quando il protocollo consente di avere la certezza che l'apparecchio dall'altro lato della linea è sempre disponibile

a ricevere, per esempio in protocolli half-duplex in cui si trasmette a turno, si può rinunciare in toto al controllo di flusso. L’errore di overr-run non dovrebbe verificarsi mai, in questo contesto, ma è bene prevedere comunque la possibilità di segnalarlo all’altro dispositivo.

La scelta per il nostro circuito.

Per il nostro circuito si era pensato inizialmente di utilizzare il controllo di flusso hardware RTS/CTS, e il circuito, anche nella versione finale, consente questa possibilità con una programmazione adeguata del microcontrollore. Una volta analizzato meglio il problema e sviluppato il protocollo, ci si è resi conto che con una progettazione attenta del protocollo non era necessario alcun controllo di flusso.

Il protocollo finale, ricalcando il più possibile fedelmente lo standard industriale SCPI nella versione implementata nello strumento Agilent 34970A, prevede semplici stringhe di testo per i comandi, terminate dal carattere new-line (ASCII LF 0x0A). Sono accettate anche righe terminate dalla combinazione ASCII CR LF (0x0C 0x0A), mentre non sono accettate righe terminate da CR da solo. La macchina è normalmente in stato di attesa di una istruzione da eseguire dal PC. Un comando riconosciuto correttamente viene eseguito non appena la riga corrente è terminata correttamente da un new-line. I comandi vengono tutti eseguiti in breve tempo, subito dopo il new-line, e non esiste alcun comando che richiede un tempo di elaborazione tale da far perdere caratteri al buffer di ricezione PIC. Subito dopo l’esecuzione del comando, viene inviata dal PIC una stringa contenente la segnalazione di comando completato oppure un messaggio di errore.

E’ proprio l’invio di caratteri sulla porta seriale l’unica elaborazione bloccante sul PIC. Mentre il PIC invia messaggi di errore o informativi, il suo buffer di ricezione può riempirsi completamente, e il firmware perdere caratteri. Non essendo previsto il controllo di flusso, è dunque demandata al programmatore del PC l’accortezza di non inviare mai due righe di testo di seguito alla macchina, senza prima aver letto la risposta o aver aspettato il tempo necessario alla macchina per inviare la risposta (se non interessa leggerla). Nel caso peggiore, ovvero per l’invio di messaggi particolarmente lungi, sono sufficienti 12 ms, ai quali vanno aggiunti, in caso di errore, ulteriori 100ms duranti i quali il PIC effettua un beep col buzzer piezoelettrico.

C’è da dire che diversi programmi di comunicazione già pronti e diverse API di programmazione arrestano automaticamente l’invio di caratteri quando si riceve qualcosa, proprio per venire incontro a periferiche half-duplex. Ad esempio il terminale di Windows

con tutta probabilità utilizza questo approccio[36], poiché anche inviando diversi comandi di seguito, con un copia/incolla, non si verificano, se non sporadicamente, problemi di buffer overrun, nonostante non sia stato previsto esplicitamente alcun controllo di flusso.

```
ROUTE:CLOSE (@111)
ROUTE:CLOSE (@112)
ROUTE:OPEN (@111)
ROUTE:CLOSE (@113)
ROUTE:OPEN (@112)
ROUTE:CLOSE (@114)
ROUTE:OPEN (@113)
ROUTE:CLOSE (@121)
ROUTE:OPEN (@114)
ROUTE:CLOSE (@122)
ROUTE:OPEN (@121)
ROUTE:CLOSE (@123)
ROUTE:OPEN (@122)
ROUTE:CLOSE (@124)
ROUTE:OPEN (@123)
ROUTE:CLOSE (@131)
ROUTE:OPEN (@124)
ROUTE:CLOSE (@132)
ROUTE:OPEN (@131)
ROUTE:CLOSE (@133)
ROUTE:OPEN (@132)
ROUTE:CLOSE (@134)
ROUTE:OPEN (@133)
ROUTE:CLOSE (@141)
ROUTE:OPEN (@134)
ROUTE:CLOSE (@142)
ROUTE:OPEN (@141)
ROUTE:CLOSE (@143)
ROUTE:OPEN (@142)
ROUTE:CLOSE (@144)
ROUTE:OPEN (@143)
ROUTE:OPEN (@144)
```

Esempio di una lunga sequenza di comandi inviata con un copia/incolla
tramite il terminale di Windows senza che ciò abbia provocato errori di buffer overrun.

Con queste scelte, si è evitato di dover implementare un buffer di ricezione software sul microcontrollore (che poteva comunque riempirsi), potendosi affidare alla sola UART hardware con i sui 2/3 byte di buffer di ricezione.

Progetto e realizzazione circuito

Giudicando sufficiente l'esperienza maturata con i circuiti di base che avrebbero costituito, come tante tessere di un puzzle, il circuito finale, si è deciso di procedere con lo schema elettrico del circuito finale. La complessità del progetto obbligava al ricorso ad un software di

[36] Nei programmi di emulazione terminale è infatti importante che i caratteri inviati al server si inframmezzino correttamente a video con quelli inseriti in locale su console.

Cad specifico per elettronica, anziché un software di disegno generico, o, peggio, carta e penna.

Eaglecad

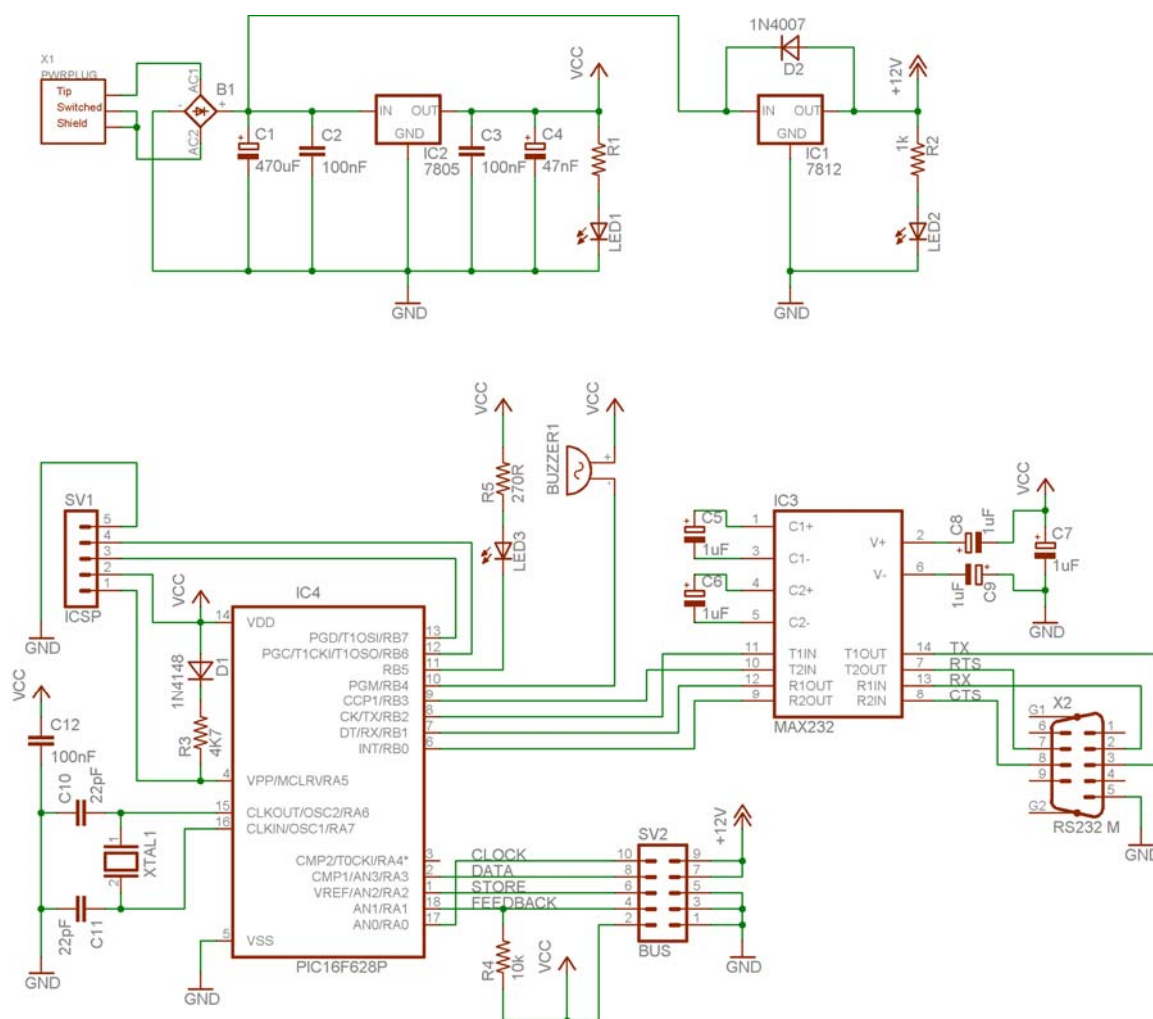
Si è scelto il software gratuito per uso personale o didattico, con alcune limitazioni sulle dimensioni della basetta, Eagle (Easily Applicable Graphical Layout Editor), della CadSoft, una software house tedesca, raggiungibile al sito http://www.cadsoft.de/.

Il cad è specifico per il disegno di schemi elettrici finalizzato al successivo sbroglio su basetta. Ha vaste librerie di componenti *through-hole* e *surface-mounted*[37], ed è molto facile estendere le librerie con i propri componenti, ammesso che si abbia a disposizione un calibro per prendere le quote sul componente reale o il data-sheet con le quote millimetriche del package. E' previsto lo sbroglio dei segnali automatico o manuale. E' in grado di esportare i percorsi utensile per plotter gerber o, con un minimo di programmazione nel linguaggio di scripting incorporato, fresatrici CNC. Il potente linguaggio di scripting interno consente di estendere o velocizzare le operazioni possibili.

Schema elettrico

Si è deciso di disegnare lo schema elettrico, ancor prima di una analisi dettagliata del protocollo di comunicazione, poiché era stato già definito tutto quanto c'era da definire. Lavorare alla progettazione del firmware, senza sapere come è fatto il circuito, sarebbe stato un lavoro inutilmente astratto e difficile.

Il circuito è diviso su due basette: controller e modulo di espansione. Sono stati disegnati separatamente gli schemi elettrici dei due circuiti.

[37] Through Hole, brevemente TH, letteralmente: "attraverso il foro", indica i componenti per il quali è necessario forare la basetta, inserire il reoforo nel foro, ed effettuare la saldatura dall'altro lato, tagliando poi la lunghezza del reoforo in eccesso. Sourface-Monted-Devices (SMD), letteralmente: "Dispositivi montati sulla superficie" sono invece quei componenti che richiedono saldature direttamente dal lato dove sono montati sulla basetta. I SMD si vanno imponendo per i costi e dimensioni ridotte, la facilità e velocità del montaggio automatizzato, e la riduzione del costo complessivo delle basette.

Schema elettrico scheda controller

Stadio di alimentazione

Il circuito doveva avere due alimentazioni: 12V per i relais, e 5V per la logica di controllo. La terza alimentazione (duale +10V -10V) viene generata direttamente dallo stesso chip che si occupa della trasmissione seriale, e non occorrevano accorgimenti particolari per essa. Lo stadio di alimentazione è collocato sul controller, e alimenta anche tutti i moduli ad esso connessi.

In alto è visibile lo stadio di stabilizzazione di tensione. Per l'alimentazione destinata ai relais, l'idea originale era di utilizzare un trasformatore da 9V, seguito da un raddrizzatore a doppia semionda con 4 diodi e un condensatore elettrolitico di livellamento (che avrebbero portato la tensione di 9V alternata a circa 11,5V continua, sia pur dotata di un ripple considerevole). Essendo il budget a disposizione ristretto, si è stati costretti ad utilizzare un trasformatore di recupero, montato nel suo contenitore con spina, che forniva 12V alternata, 1,58A.

AC/AC Adapter di recupero

Al trasformatore è stato cambiato il connettore originale utilizzandone un altro di più ampia diffusione, idoneo ad essere innestato in quello montato poi sulla basetta. Era necessario abbassare la tensione raddrizzata e livellata prelevata dal trasformatore; abbandonata l'idea di poter dimensionare correttamente tutto lo stadio di alimentazione in base alle condizioni di utilizzo, si è deciso di utilizzare un regolatore integrato a 3 piedini, il 7812, così da consentire di alimentare il circuito con qualsiasi tensione continua da 15V a 18V o alternata da 10V a 15V. Se è vero che i relais assorbono 100mA ciascuno, e i led accanto ai relais 15mA ciascuno, 16 relais accesi, con i relativi led, assorbono 1,8A. Il nostro trasformatore è da 1,58A, quindi appena insufficiente. Infatti accendendo 15 o 16 relais a volte la tensione di alimentazione cade così tanto da far spegnere e riaccendere la parte logica del circuito. Basta utilizzare un alimentatore adeguato al carico per non osservare più il malfunzionamento. C'è da dire che la condizione d'uso con tutti i relais accesi non ha molto senso in quanto collega tutti gli ingressi a tutte le uscite contemporaneamente, e non ha utilità pratica alcuna. Tipicamente si lavorerà con 4-8 relais inseriti per ciascun modulo, quindi anche con l'alimentatore attuale il circuito può utilizzare 4 moduli in condizioni operative tipiche.

Anche la scelta del 7812 può sembrare non felice, poiché la corrente massima erogabile, per la quale è garantita la regolazione della tensione è 1A, ma le prove pratiche dimostrano che per i relais, che non hanno bisogno di tensione stabilizzata, il 7812 funziona egregiamente anche a quasi 2A. Altrimenti basta sostituirlo con un regolatore da 3A con stessa piedinatura e stesso ingombro.

Il dimensionamento del dissipatore pure è stato sommario. Sapendo che gli integrati della serie 78XX sono protetti dal surriscaldamento, si può partire con un dissipatore e procedere per tentativi: se scalda troppo, se ne monta uno più grosso. E' possibile, ed anche semplice, effettuare calcoli teorici sulla potenza da dissipare, e valutare la resistenza termica del

dissipatore necessario a garantire una temperatura del case e una temperatura alla giunzione dei semiconduttori, ma questi conti lasciano il tempo che trovano, poiché il coefficiente di convezione naturale per il dissipatore non è mai noto con precisione, e occorre comunque una prova pratica: tanto vale farla direttamente.

Concludendo, l'interesse primario è stato dato a questioni più interessanti del dimensionamento dell'alimentazione, e per onorare il risparmio, lo stadio di alimentazione è leggermente sottodimensionato, ma perfettamente funzionante in condizioni tipiche. In caso di sviluppi futuri del progetto, si può decidere di preferire il dimensionamento di caso peggiore e ridisegnare la basetta.

Il led LED2 segnala che l'alimentazione è presente, e dà un rapido riscontro visivo, con l'affievolimento, se il 7812 entra in protezione riducendo la corrente erogata. Il 7812 è stato protetto con il diodo poiché i relais producono extra-correnti di apertura considerevoli, che potrebbero distruggere l'integrato se riescono a farsi strada dai moduli fino alla scheda del controller, senza riuscire a scaricarsi nei diodi di ricircolo dell'ULN2803. Non sono stati inseriti condensatori poiché non interessa realizzare una buona stabilizzazione per i relais.

Lo stadio di alimentazione per i 5V è costruito attorno ad un 7805. Questo è circondato da condensatori, seguendo i consigli del data sheet del componente, per assicurare un'ottima regolazione di linea e di carico per la logica. L'evento di spegnimento del microcontrollore per alimentazione insufficiente è infatti catastrofico: alla riaccensione rieseguirebbe il firmware dal principio, perdendo tutte le configurazioni, e occorre estrema cura per evitare questo fenomeno. Il condensatore C12, collegato molto prossimo ai piedini di alimentazione del microcontrollore, fa fronte ai tipici assorbimenti impulsivi di corrente dei chip in tecnologia CMOS, bypassando l'alimentazione proprio vicino all'integrato. Il led LED1 segnala che l'alimentazione a 5V è presente.

Microcontrollore

Il resto del circuito è tutto costruito attorno al microcontrollore.

Sul lato sinistro spicca il **cristallo di quarzo**. E' stato scelto un oscillatore al quarzo poiché è necessaria una temporizzazione precisa per realizzare la comunicazione seriale in maniera affidabile. Il valore dei condensatori verso massa presenti accanto al quarzo è consigliato direttamente sul data sheet.

Sul lato sinistro spicca pure un **header a 5 pin utilizzato per l'ICSP** (In circuit serial programming). D1 e R3 servono ad evitare che la tensione di 13,8V, applicata al piedino 4 del PIC durante la riprogrammazione, possa raggiungere l'alimentazione a 5V del circuito,

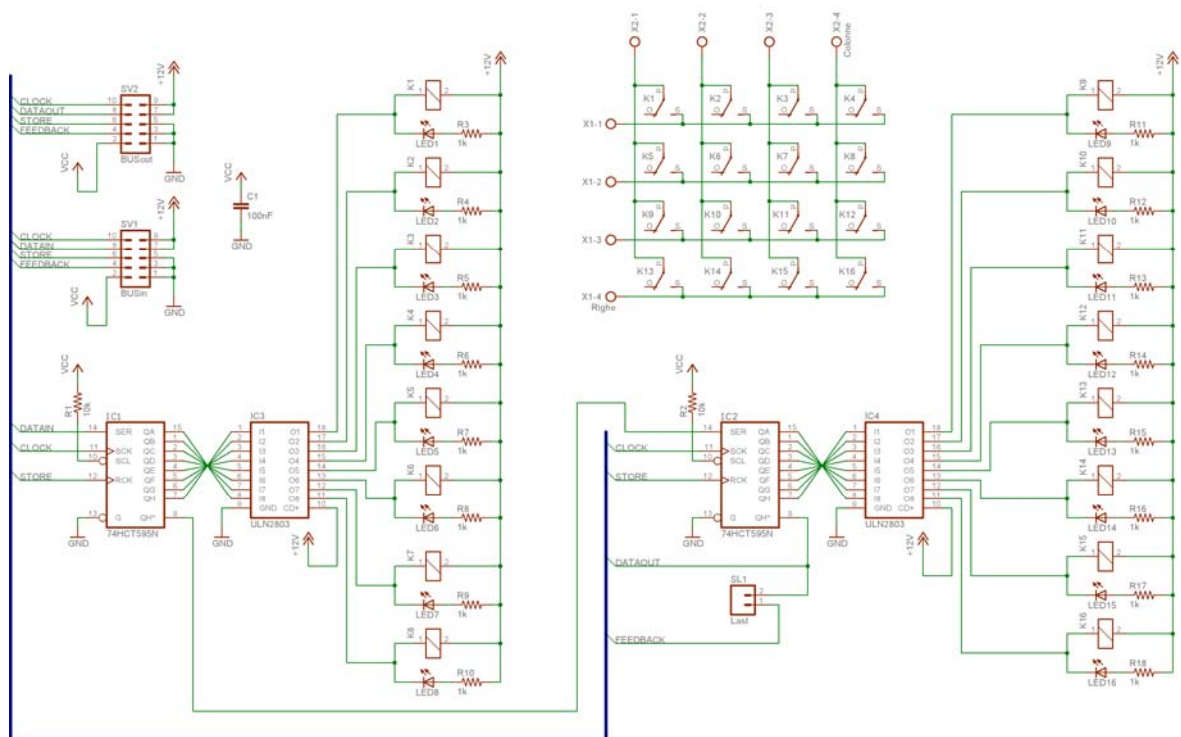
danneggiando il PIC e gli altri integrati montati sul circuito. Durante l'uso normale, lo stesso diodo e la stessa resistenza realizzano un pull-up a 5V positivi sul piedino 4 (MCLR/), che è un segnale di reset attivo basso per il PIC, a meno che non venga disabilitato. Il PIC scelto, PIC16F628, consente la disabilitazione del segnale, altri PIC pin-to-pin compatibili, necessitano invece del pull-up verso 5V. Si è deciso pertanto di montare sul circuito D1 e R3 in modo da consentire l'eventuale successiva migrazione verso PIC differenti con la stessa piedinatura, che abbiano bisogno di tale pull-up.

A due pin del PIC sono collegati il **LED** LED3 e il **Buzzer piezoelettrico** BUZZER1, utilizzati per la segnalazione di errori. I pin del microcontrollore sono quasi tutti utilizzabili come porte I/O generiche, la scelta di quali pin utilizzare è stata fatta tenendo conto dello sbroglio sulla basetta.

In basso è presente l'header a 10 pin del **bus di sistema**, dello stesso tipo di quello usato per il JTAG delle FPGA, che serve a collegare il controller ai moduli. Feedback è configurato come ingresso nel microcontrollore, e ha una resistenza di pull-up. Su questa linea tornano i bit uscenti dall'ultimo shift-register, sull'ultimo modulo in cui sia stato inserito il ponticello di terminazione. La resistenza di pull-up garantisce una lettura obbligata in caso di feedback non collegato a nulla (flottante).

Gli altri tre segnali, clock, data, store, sono collegati a piedini configurati come uscite, che piloteranno la catena di shift-register. Inizialmente il segnale di clock era collegato al pin 3 del PIC, ci si è poi resi conto, solo dopo aver montato la basetta, che tale pin ha una uscita del tipo open drain anziché push-pull. Si poteva inserire una resistenza di pull-up sul circuito, ma dato lo sbroglio effettuato e la mancanza di linee a +5V lì vicino, è risultato più agevole spostare il collegamento dal pin 3 al pin 17. E' questo un esempio d'uso delle tecniche *cut & wire* per errori di cui ci si rende conto solo dopo aver realizzato la basetta. L'errore viene poi riportato sullo schema elettrico e sul disegno del PCB, in modo che le successive basette non lo contengano più.

Le altre linee del connettore per il bus di sistema sono di alimentazione: tre per il riferimento di massa, due per l'alimentazione a 12V, una per l'alimentazione a 5V.

Infine 4 pin del PIC sono collegati al **MAX232**, un adattatore di livello da TTL 0/5V a RS-232 +10V/-10V. Il MAX232 incorpora un duplicatore e un invertitore di tensione, in grado di generare le tensioni di +10V e -10V direttamente dall'alimentazione a 5V, richiedendo solo 5 condensatori elettrolitici esterni. Il connettore seriale è collegato con il pinout tipico di dispositivi DTE.

Schema elettrico modulo con relais

A fronte di uno sbroglio e un montaggio che richiedono qualche attenzione, lo schema elettrico della basetta per i moduli a relais è invero piuttosto semplice.

Sono presenti due header a 10 pin per il **bus di sistema**: uno di ingresso e uno di uscita, etichettati BUSin e BUSout. Tutti i segnali sono collegati in parallelo tra i due connettori, tranne il segnale Data, che viene prelevato da BUSin, e inviato al primo shift-register. L'uscita QH' del primo shift register viene inviata all'ingresso data del secondo shift-register. L'uscita QH' di quest'ultimo viene inviata al segnale Data di BUSout. Si realizza così il collegamento in cascata degli shift-register ospitati sui vari moduli collegati al sistema. Il dataout può essere circuitato con feedback con un apposito ponticello. Questo è il ponticello di terminazione, da inserire solo sull'ultimo modulo della catena, che consente al controller di controllare la funzionalità degli shift-register e di determinare quanti moduli sono collegati contando dopo quanti colpi di clock un pattern di bit inviato agli shift-register percorre l'intera catena e torna sul piedino di feedback.

Il **condensatore C1** effettua il bypass dell'alimentazione a 5V, e andrà collocato, in fase di sbroglio, quanto più vicino possibile agli shift-register. E' buona norma prevedere questi condensatori vicino ad ogni circuito integrato logico CMOS, che assorbe corrente rilevante dalle alimentazioni solo quando qualche stadio interno commuta, quindi in maniera impulsiva.

I segnali di output enable e reset sono collegati fissi alle alimentazioni, come descritto prima nel paragrafo relativo al funzionamento degli shift-register.
Le uscite degli shift-register pilotano gli ingressi degli ULN2803, specificatamente pensati per collegare carichi induttivi a integrati logici. Le connessioni appaiono aggrovigliate poiché la piedinatura dei due integrati consente uno sbroglio immediato con questo tipo di collegamento: cioè che appare aggrovigliato sullo schema elettrico è in realtà un collegamento dritto sulla basetta: osservare i numeri dei piedini per rendersene conto.
Il pin 10 degli ULN2803 fa capo ai diodi di clamping, e va quindi collegato ai 12V. Le uscite open drain dell'ULN2803 sono collegate ciascuna ad una coppia relais/led. Ciascun led ha la sua resistenza limitatrice di corrente in serie.
Gli interruttori dei relais, anziché essere stati disegnati accanto ai relativi relais, sono stati raggruppati tutti insieme per evidenziare il collegamento a matrice che si desiderava realizzare. Le righe e le colonne della matrice sono state rese disponibili all'utilizzatore mediante morsetti a vite.

Sbroglio

Disegnato un circuito elettrico, si procede all'operazione di sbroglio, ovvero a sistemare i componenti su una basetta e collegarli con piste di rame che rispettino la stessa topologia dei collegamenti dello schema elettrico. Con i metodi di preparazione delle basette artigianali, è preferibile uno sbroglio monofaccia, vale a dire componenti da un lato, piste di rame dall'altro lato della basetta. La topologia insegna che così procedendo è necessario aggiungere dei ponticelli, in grado di scavalcare alcune piste, sul lato componenti, oppure anche direttamente sul lato piste, rendendo i ponticelli un po' bruttini dal lato piste, ma invisibili dal lato componenti.
L'operazione di sbroglio non è banale. Occorre tenere presente che le piste di alimentazione devono avere larghezza sufficiente in base alla corrente che ci deve scorrere dentro, per evitare pericolose cadute. Inoltre è bene diramarsi a stella anziché in cascata dallo stadio di alimentazione a tutti gli utilizzatori, in modo da garantire, per quanto possibile, la stessa tensione ovunque. Alcune piste di alimentazione particolarmente critiche possono essere ingrossate ricoprendole integralmente di stagno in fase di saldatura.
Due piste parallele e poco distanziate si comporteranno come un condensatore, e segnali elettrici possono copiarsi da una all'altra per accoppiamento capacitivo. Le piste con i segnali di programmazione è bene che abbiano tutte la stessa lunghezza. Alcuni componenti vanno posizionati molto vicino ad altri, per esempio i condensatori di bypass dell'alimentazione

vicino agli integrati CMOS. Occorre prevedere gli ingombri dei dissipatori da montare sui componenti che ne hanno bisogno. Se il circuito prevede led di segnalazione o interruttori, questi devono poter essere visibili o accessibili, e quindi non vanno montati vicino a componenti con ingombro verticale elevato.

A volte si può rivedere lo schema elettrico per facilitare lo sbroglio: è il caso dei piedini di I/O digitale dei microcontrollori, che possono essere riassegnati ad altri senza particolari problemi.

Eagle facilita di molto il compito, poiché per ogni componente inserito nello schema elettrico, viene automaticamente generato anche il *footprint* sulla basetta, con *airwires*[38] (rappresentati come linee gialle scure da Eagle) che indicano quali pin sono collegati fra loro: è più facile così posizionare correttamente i componenti prima di iniziare lo sbroglio. Alcuni componenti, come le resistenze, possono essere montati egualmente bene in verticale con i due fori molto vicini fra loro (per risparmiare spazio), oppure in orizzontale, con i due fori più distanti, per agevolare lo sbroglio facendo passare altre piste tra un foro e l'altro. Eagle consente di sostituire al volo il package di un componente con un altro, in fase di sbroglio, senza modificare minimamente lo schema elettrico.

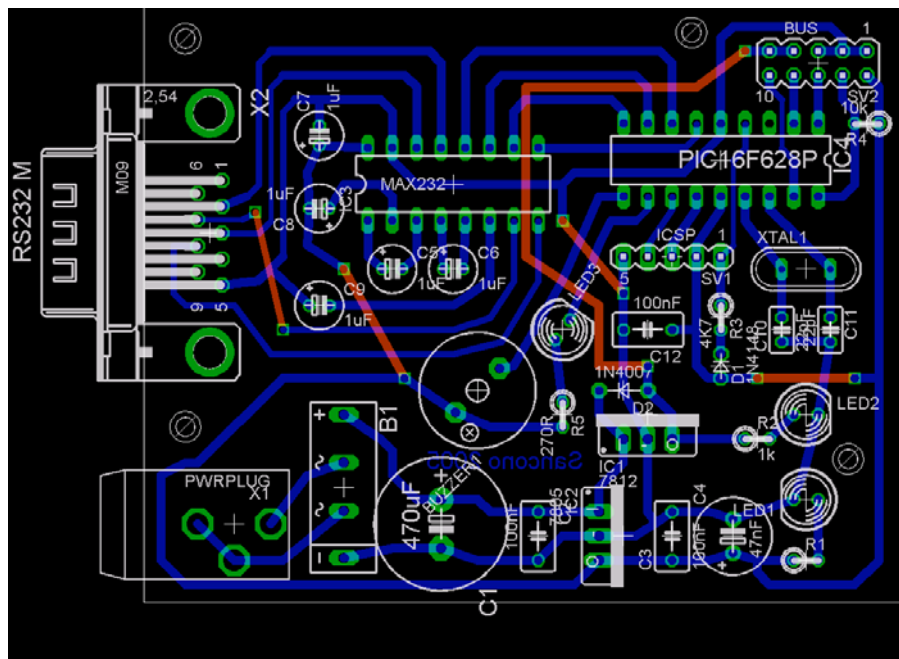
E' spesso necessario lasciare dello spazio attorno ai componenti da cui si dipartono più piste, per consentire a queste di distanziarsi tra loro e dirigersi verso i punti di destinazione. Solo l'esperienza insegna quanto spazio lasciare accanto ad ogni componente, cercando l'equilibrio tra non sprecare spazio e consentire comunque uno sbroglio agevole.

Occorre inoltre prevedere lo spazio per i fori di fissaggio e per le teste dei distanziatori che verranno utilizzati per montare la basetta: se i distanziatori sono metallici, occorre lasciare dello spazio attorno ai fori libero da piste per evitare pericolosi corto circuiti, realizzati se poi i distanziatori, ciascuno in contatto con una pista diversa, vengono poi fissati ad un'unica carcassa metallica. Utilizzando distanziatori plastici si evita questo problema.

Eagle dispone di un *autorouter*, vale a dire un algoritmo che genera da solo le piste, ma è per lo più adatto a basette doppia faccia, in cui le piste possono passare da un lato all'altro attraverso fori in cui viene depositato del metallo. Questa tecnica è disponibile solo con procedimenti industriali, ed è fuori dalla portata del nostro laboratorio. Per lo sbroglio monofaccia, è di solito preferibile uno sbroglio manuale, almeno per le piste di alimentazione, che devono rispondere a regole ben precise, difficili da insegnare ad un PC. L'*autorouter* può essere ancora utilizzato per mostrare la fattibilità dello sbroglio: se l'*autorouter* riesce a collegare il 90% delle piste senza ponticelli, presumibilmente uno sbroglio manuale

---

[38] Trad. letterale: "Fili volanti"

collegherà l’80% delle piste, ma molto meglio, vale a dire evitando tortuosi cammini per piste che rischiano di rompersi o di andare in corto con altre vicine. Se l’*autorouter* non arriva al 90%, è bene chiedersi se non è il caso di riposizionare i componenti prima di procedere allo sbroglio.

Sbroglio basetta controller

In figura lo sbroglio della basetta del controller. Il disegno è visto dall’alto, dal lato componenti. Gli ingombri dei componenti sono indicati in bianco, come pure le sigle e i valori. In verde i PAD, ovvero le piazzole di rame sulle quali verranno saldati i reofori dei componenti. In blu le piste di rame. Siccome la basetta è vista dall’alto dal lato dei componenti, tanto le piste che le piazzole vanno immaginate come se si vedessero in trasparenza attraverso la basetta. Una volta riportate sul rame, queste saranno allo specchio se osservate dal lato piste. In rosso Eagle indica le piste presenti sul lato componenti di una basetta doppia faccia. Il nostro circuito verrà realizzato su basetta monofaccia, sono stati utilizzati quindi le piste rosse di Eagle per indicare i ponticelli da realizzare sulla basetta. Si notino anche i fori di fissaggio per i distanziatori. Non sono agli angoli perché con la scelta fatta si è potuta usare una basetta più piccola. LED3 e BUZZER1 sono stati ruotati di un angolo arbitrario per agevolare lo sbroglio. Si è cercato di mantenere i ponticelli corti e rettilinei, ma in un caso non vi si è riusciti.

La totalità dei componenti utilizzati era già presente nelle librerie di Eagle. Fa eccezione il connettore di alimentazione, POWERPLUG, che è stato disegnato utilizzando un calibro per prendere le quote sul componente reale.

Finito lo sbroglio è una buona idea lanciare il tool DRC (Desgin rules check) che controlla le distanze reciproche tra gli elementi per verificare che siano superiori ai minimi impostati per la propria tecnologia costruttiva. I parametri del DRC di Eagle sono stati calibrati per il metodo di incisione utilizzato, in modo da trarre proficui suggerimenti da questo tool.

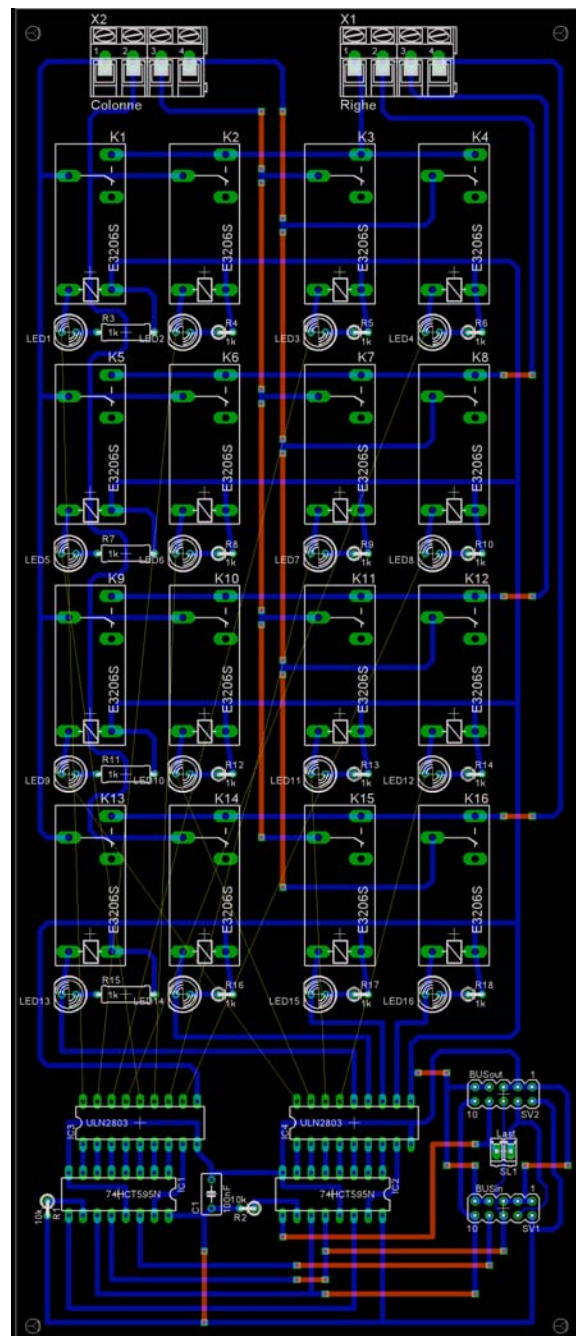
La basetta dei moduli di relais prevede molti più ponticelli, poiché è impossibile realizzare uno sbroglio monofaccia efficace per la topologia del circuito, infatti:

- Gli interruttori dei relais sono collegati a matrice di righe e colonne, e si capisce che i relativi segnali finiscono per incrociarsi reciprocamente più volte;
- Diversi elementi hanno collegamenti in parallelo, pin-to-pin, che sono impossibili da sbrogliare su un circuito monofaccia, vale a dire gli header del bus di sistema e gli shift register, e l'alimentazione 12V, che deve interessare tutti i gruppi relais + led + resistenza;
- I relais vanno collegati ciascuno alla relativa uscita dell'ULN2803, ma questo collegamento a raggiera male si concilia con il collegamento a matrice e con il collegamento dell'alimentazione 12V.

Si è fatto il possibile, collegando senza ponticelli l'alimentazione 12V, collegata in orizzontale riga per riga ad una mandata principale verticale sul lato destro. Sono stati collegati senza ponticelli i tratti orizzontali delle righe delle matrici. Per i tratti verticali, si è riusciti comunque a mandare altri due segnali delle colonne delle matrici, uno dal lato destro (il sinistro era impegnato dall'alimentazione), e l'altro passando abilmente tra i piedini delle resistenze limitatrici dei LED.

Gli altri due tratti verticali delle colonne si risolvono in una serie di ponticelli con il loro spazio dedicato a metà basetta. Ponticelli sul lato destro collegano le righe: i ponticelli sono necessari per scavalcare l'alimentazione, già mandata dallo stesso lato.

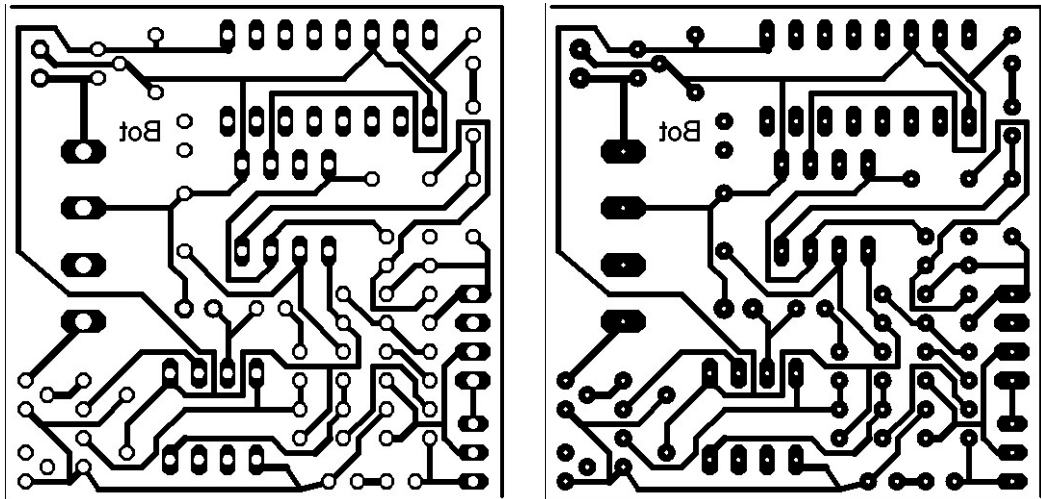
Il collegamento tra gli shift-register e gli ULN2803 è diretto, grazie all'intreccio presente invece nello schema elettrico. Resta una marea di ponticelli, e di spazio occupato, sul lato inferiore della basetta, per collegare tra loro i piedini omologhi dei due shift-register e dei due header per il bus di sistema. Infine è stato possibile collegare solo la riga inferiore di relais alle uscite degli ULN2803. Gli altri 12 collegamenti sono stati realizzati con fili dal lato piste della basetta.

L'alternativa era occupare ancora più spazio in basetta (la basetta ha già dimensioni di 25 x 11,5 cm, più che considerevoli[39]), e aumentare moltissimo il numero di ponticelli.

Sbroglio basetta modulo relais

Incisione basette

pshouse-drill-aid: uno script per Eagle rilasciato con GPL

[39] Nelle basette molto grandi insorge la difficoltà di garantire una incisione uniforme su tutta la superficie, con i pericoli di piste assottigliate, o interrotte, in una zona, e piste ingrossate, o in corto, in un'altra zona.

Eagle può stampare direttamente i master per i PCB oppure esportare percorsi utensile per frese CNC, files per plotter Gerber, o immagini da stampare con altri software di grafica. Le immagini prodotte da Eagle sono perfette per procedimenti di incisione industriale, ma presentano qualche inconveniente per l'incisione artigianale.

Master esportato da Eagle e master esportato dallo script di mia ideazione

Si osservi l'immagine di sinistra:

- Lo spazio interno delle piazzole è esattamente uguale al diametro del foro da realizzare. Questo significa che bisogna andare col trapano al posto giusto, e non si incontrerà rame nel percorso. Può essere difficile se non impossibile la centratura senza alcuna guida su materiale tenero, e se si sbaglia all'inizio, si è perduti, perché è pressoché impossibile correggere un foro iniziato nel posto sbagliato. Inoltre il procedimento di incisione casalingo può mangiare un po' il diametro interno, e lo spazio senza rame diventerà addirittura più grande del foro da fare.
- Diversi componenti di libreria standard hanno piazzole con fori da 0,6 mm (per esempio i resistori). Un hobbysta di solito preferisce fori appena più grossi (0,8 millimetri) ottenendo di usare una punta meno delicata, e di infilare più facilmente i reofori in un foro appena più largo. Chiaramente forando con la punta da 0,8 millimetri una piazzola pensata per un foro da 0,6 millimetri, di questa non resterà traccia, perché volerà via non appena si avvicina il trapano. Eagle non consente di modificare facilmente il diametro dei fori dei pad dei componenti: l'unica strada è quella di modificare i componenti nella libreria, ma può essere faticoso, noioso e lungo.

- I fori non legati a pad o via, per esempio quelli di montaggio della basetta, normalmente non vengono riportati affatto sul rame, poiché sono nel layer a parte holes e non nei layer pad/via/top/bottom, che sono i soli riportati su rame.

Per risolvere questi problemi, ho sfruttato il linguaggio di scripting interno di Eagle e approntato uno script chiamato pshouse-drill-aid, che distribuisco con GPL[40]. Visitare il mio sito http://www.ideegeniali.it/, sezione progetti, per maggiori informazioni riguardo al modo di operare dello script o per scaricare l'ultima versione rilasciata.

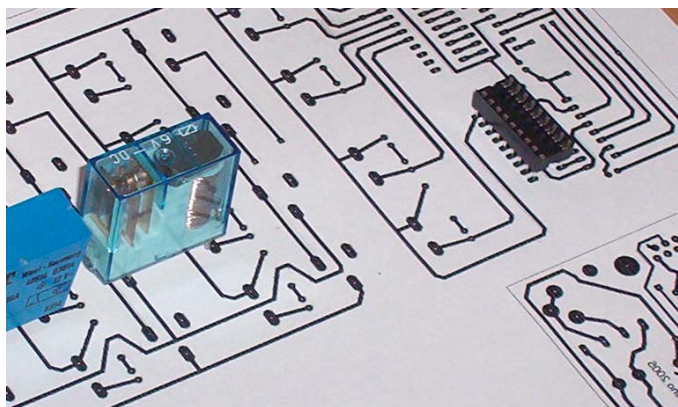
Il risultato è visibile nella figura di destra, i vantaggi sono notevoli:

- I diametri interni di tutti i pad e via, di qualunque forma fossero, sono stati ridotti, in modo da facilitare la centratura del trapano, che avverrà su metallo, non sul tenero. La punta verrà naturalmente guidata verso il centro del foro e, se proprio si sbaglia, il rame è più resistente della basetta rimasta scoperta, e se ci si accorge subito di aver sbagliato, si fa in tempo a deviare la punta verso il centro della piazzola, dove il minuscolo cerchietto lasciato senza rame la catturerà nella posizione corretta
- Lo script ha un comportamento differente a seconda della forma dei pad: I pad per i resistori, che comprendono fori da 0.6 millimetri nella libreria standard, e che verrebbero mangiati via completamente da un foro da 0,8 millimetri, ora sono stati allargati e possono essere tranquillamente forati a 0,8 millimetri senza paura di farli volare via. I pad di forma allungata (quelli dei circuiti integrati) non sono stati allargati, per evitare corto-circuiti con le frequenti piste che passano tra un pin e l'altro dei circuiti integrati. Questi pad sono infatti più che adeguati così come sono. Anche per i pad allungati è stato però ristretto il foro centrale per facilitare la centratura.
- Sono stati generati inoltre cerchietti per i fori di fissaggio, in modo da riportare sul rame dove vanno effettuati.

Stampa di prova

Prima di incidere una basetta, è una buona idea realizzare una stampa di prova su un comune foglio di carta, in modo da verificare gli ingombri dei componenti (si verifica che possano essere sistemati agilmente uno accanto all'altro) e il passo tra i piedini, che deve essere esattamente uguale a quello dei componenti.

---

[40] GPL = GNU General Public License. E' una licenza che consente l'utilizzo gratuito del programma, consente la ridistribuzione, consente la realizzazione di lavori derivati, imponendo le condizioni che la ridistribuzione sia libera e gratuita, e che i lavori derivati conservino l'informazione sull'autore originale e vengano distribuiti con la stessa licenza.

Stampa di prova e verifica dimensioni corrette dei *footprint*

Successivamente si procede al taglio delle basette nelle dimensioni desiderate. Ad una fiera di elettronica sono state comprate a buon prezzo delle strisce di basette doppia faccia, sfridi di produzione, con un rapporto di forma a dir poco particolare: 12 cm x 200 cm. Sono state tagliate a misura leggermente abbondante (per tolleranze di centratura del master nel mio procedimento di incisione) con un disco da taglio montato sul trapanino, dopo aver sistemato la basetta su sostegni di polistirolo[41]: il polistirolo è contemporaneamente solido da mantenere la forma, cedevole da adattarsi alla forma del materiale, elastico da assorbire le vibrazioni.

Taglio delle basette a misura leggermente abbondante con i dischi da taglio per trapanino

Dopo il taglio delle basette, si procede ad una accurata pulizia con pasta abrasiva: alcuni suggeriscono la pagliette metalliche in vendita presso i ferramenta, che rendono molto lucido il rame, io suggerisco del comune Vim in polvere. E’ importante rimuovere impronte digitali

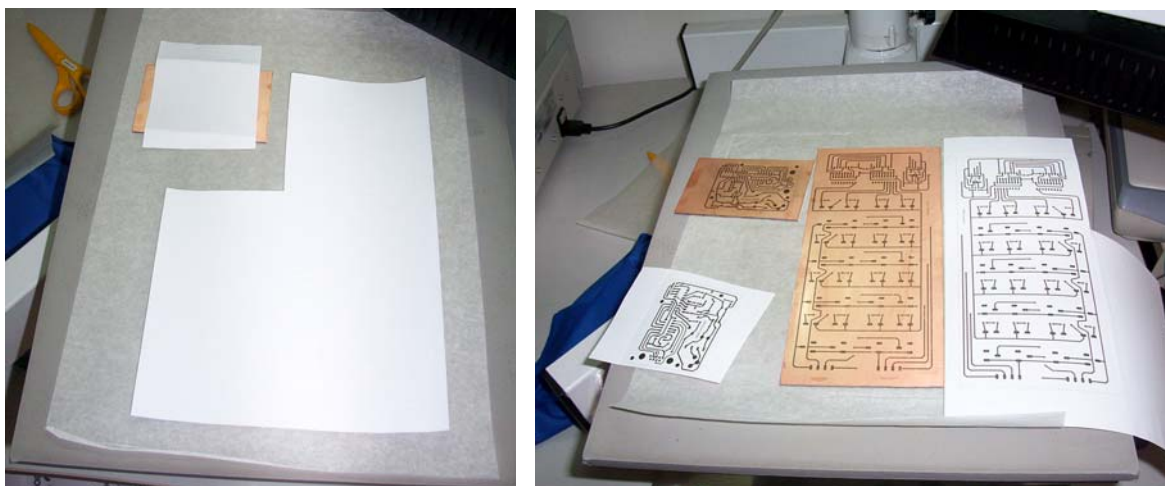
---

Questo tipo di distribuzione favorisce lo scambio di informazioni tra programmatori e lo sviluppo a più mani di programmi validi in tempi brevi.

e asperità della superficie, per garantire risultati ottimali nelle fasi successive del procedimento di incisione.

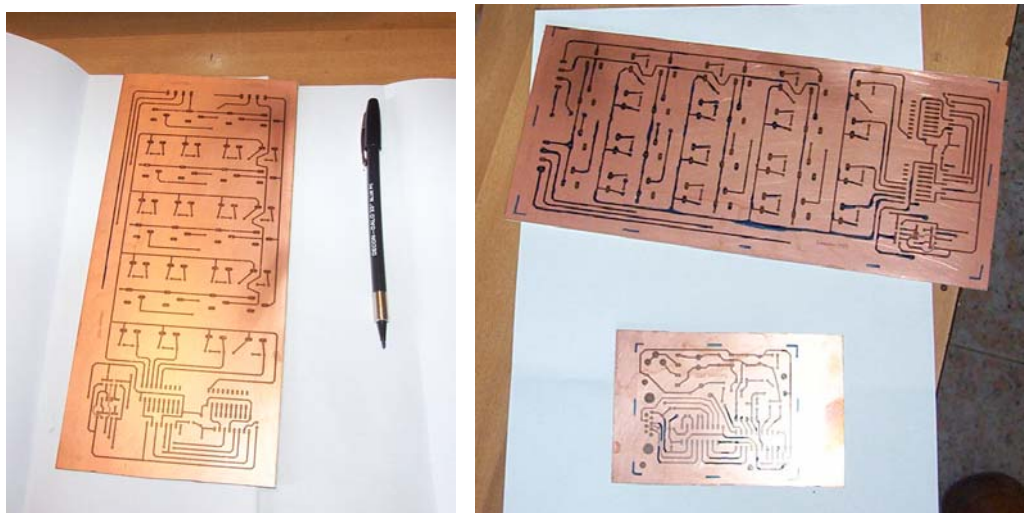
Pulitura con polvere abrasiva del lato rame della basetta.

Tra la lucidatura della basetta e il successivo riporto delle piste, è bene non passi molto tempo, perché il rame si ossida molto rapidamente e lo strato di ossido può rendere più difficile l'aderenza al toner nel passaggio successivo.

[41] Vedere sul mio sito http://www.ideegeniali.it/, sezione progetti, il progetto documentato di un traforo a filo caldo per polistirolo, utilizzato per tagliare i sostegni.

Il metodo “stira e ammira” in azione. Riporto della stampa laser a colori sulla basetta tramite pressa a caldo.

Il cuore del metodo “stira e ammira” è il riporto delle piste senza ricorrere al bromografo, né tanto meno a fotoresist, soda caustica, o basette presensibilizzate. E’ sufficiente effettuare la stampa delle piste, a specchio su un comune foglio di carta purché non molto poroso con una stampante laser a colori, che utilizza un toner dall’aspetto ceroso, facile da fondere nuovamente.

Appena stampato, il foglio di carta va immediatamente rovesciato sulle basette in rame, cercando di centrarlo il più possibile. Se disponibile, usare una pressa a caldo per uso tipografico per trasferire il toner, altrimenti un comune ferro da stiro regolato in posizione fibre sintetiche, avendo cura di preriscaldare la basetta e di non dimenticare nessuna zona del circuito.

Il toner è un inchiostro in polvere, fissato a caldo. Riscaldandolo nuovamente, una parte si staccherà dal foglio e si trasferirà sulla basetta. Il toner non è solubile in acqua, quindi non viene attaccato dal cloruro ferrico, o da altri composti per incisione solubili in acqua, al pari del fotoresist.

Correzione imperfezioni con pennarello acrilico

Siccome è purtroppo passato del tempo tra la stampa e il trasferimento con pressa a caldo, poiché ci si era dimenticati di accendere la pressa per tempo, e date le generose dimensioni della basetta (più difficile portarla tutta a temperatura e garantire un processo uniforme su tutta la superficie), il toner si era ben fissato alla carta, si era seccato parecchio, e si è screpolato in qualche punto durante il trasferimento. Incidere la basetta sarebbe significato sprecarla, poiché in corrispondenza delle spaccature del toner si sarebbero ottenute piste interrotte non più utilizzabili.

E' stato sufficiente qualche ritocco con una penna decon-dalo, ad inchiostro acrilico, specifica per il disegno manuale delle basette, per ovviare all'inconveniente. Anche se si è stati sfortunati è c'è stato l'inconveniente di dover ripassare le piste screpolate col pennarello (di solito non succede), si è comunque risparmiato tempo, materiali, attrezzature (a patto di avere una copisteria vicino casa), denaro, con questo metodo, rispetto al classico metodo di fotoriporto con bromografo.

Cloruro ferrico in grani

Per l’incisione si è usato il più che classico, docile e tranquillo[42] cloruro ferrico. Si è scelto il tipo in grani, da sciogliere in comune acqua di rubinetto, anziché liquido già dosato, poiché così è possibile regolare a proprio piacimento la concentrazione per ottenere tempi di incisione più o meno rapidi. Personalmente preferisco tempi di incisione di 12 minuti circa, così che sbagliando di un minuto in più o in meno la basetta è comunque utilizzabile. Nel caso si debbano produrre molte basette identiche, si possono preferire tempi di incisione più rapidi (4-5 minuti), ottenibili con una concentrazione maggiore o portando la soluzione a temperatura più alta, ma occorre essere molto precisi nell’istante di arresto reazione per non assottigliare le piste più piccole fino a distruggerle. Il cloruro agisce infatti inizialmente verticalmente, ma una volta che sia stato corroso tutto il rame esposto, comincia ad agire lateralmente attaccando le piste ai lati, passando sotto l’inchiostro.

Anziché travasare continuamente il cloruro ferrico da una vaschetta ad una bottiglia e viceversa, con tutti i problemi che possono derivare[43], si è preferito conservarlo sempre nella “sua” vaschetta. L’unica accortezza è stata quella di sceglierne una con chiusura ermetica. Man mano che il cloruro ferrico perde di efficacia, aggiungo semplicemente altri grani, ritardando di molto il momento in cui è necessario preparare una nuova soluzione, poiché lo smaltimento è difficoltoso e occorre rivolgersi a centri specializzati[44].

[42] Esistono altri composti per l’incisione che richiedono tempi di incisione più rapidi, ma comportano reazioni più violente e qualche volta esotermiche.

[43] Il cloruro ferrico macchia molto tenacemente, e corrode i metalli teneri, come ad esempio i tubi di rame degli impianti fognari, è bene quindi che non sgoccioli in giro.

[44] Non si può buttare il cloruro ferrico nelle fogne perché corrode i tubi, è inquinante, è acido. Per il corretto smaltimento, si procede, per grandi linee, così: occorre prima neutralizzare l’acido aggiungendo, lentamente, una base in soluzione, quindi far evaporare le parti volatili

Sospensione a filo della superficie e appena sotto.

Le basette che avevo a disposizione, sfridi di produzione avuti per pochi centesimi, erano doppia faccia. Occorreva pertanto rimuovere completamente il rame di una delle due facce, che altrimenti provocherebbe un corto circuito generale tra tutti i componenti. Si può procedere per via meccanica, asportandolo con frese o platorelli di carta vetrata, ma così si ottiene tanta polvere (tollerabile in una officina, un po’ meno in una casa privata o in un laboratorio universitario) e una basetta dalla superficie rugosa che non consente di prendere appunti con una penna indelebile vicino ai componenti, poiché l’inchiostro si spande nelle microfessure. Si è preferito allora rimuovere per via chimica il lato di rame che non interessa.

Il metodo classico, che consente di velocizzare i tempi di incisione, prevede di agitare continuamente la basetta, o, ancora meglio, di insufflare aria tramite una pompa e rimescolare continuamente la soluzione. La difficoltà è rendere questo processo uniforme su tutta la superficie: agitando la basetta manualmente, infatti, quello che si ottiene è una incisione più rapida ai bordi rispetto al centro della scheda, poiché il rame che passa in soluzione riduce localmente la velocità della reazione: ai bordi, a causa del rimescolio più pronunciato, avendo più cloruro ferrico fresco (senza rame in soluzione), la reazione è più rapida.

Al continuo agitare uniforme, preferisco personalmente la soluzione opposta: calma piatta. La basetta è stata sospesa con due fili di cotone, fissati da un lato con colla a caldo sul piano di appoggio, dall’altro ancorati a due pesi, in figura due portapenne, così da consentire di regolare le altezze. Il cotone si lascia attraversare dal cloruro ferrico, ed è necessario spostare solo una o due volte la posizione dei fili per non trovarsi con delle sottili linee non attaccate. Il rame passa in soluzione, staccandosi dalla basetta, e raggiunge il fondo del recipiente. La

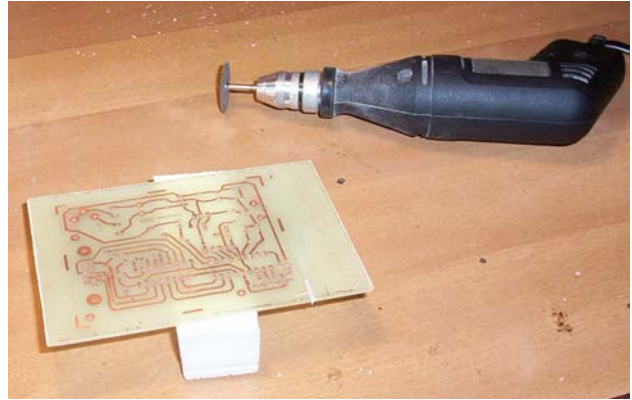
---

aumentando la concentrazione, infine impastare con polveri cementanti e conservare il “mattone” così ottenuto in appositi centri. Più indicato rivolgersi a ditte specializzate.

sostituzione con cloruro fresco è lenta, ma costante e uniforme su tutta la superficie. A costo di aspettare 5 minuti in più, si ottiene però un risultato certo e migliore.

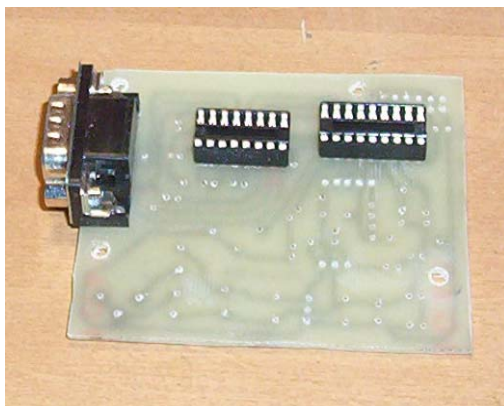
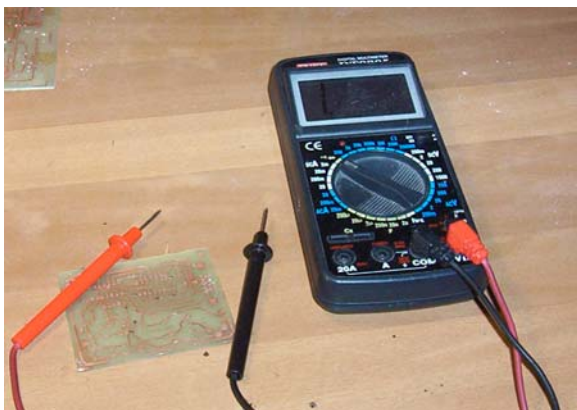
Si noti in figura che la basetta è sospesa a filo di superficie per incidere la faccia di non interesse, e appena sotto la superficie per incidere la faccia di interesse: è così possibile controllare in trasparenza il procedere dell'operazione e l'eventuale presenza di dannose bolle d'aria sotto la superficie, evitando di dover continuamente togliere e rimettere la basetta in posizione per un raffronto visivo diretto.

Acqua per arrestare subito la reazione (meglio vaschetta che lavandino).
Trielina per rimuovere il toner e l'acrilico.

Finita l'incisione, si getta la basetta, dopo averla fatta sgocciolare, in una vaschetta contenente acqua, per arrestare immediatamente la reazione. Si finisce quindi di sciacquare la basetta in un lavandino, quando la concentrazione di cloruro è diventata davvero minima. E' necessario infine rimuovere toner e acrilico. Si può usare la trielina, in grado di rimuovere entrambi contemporaneamente, o altri solventi adatti. Il rame appare ben lucido e pulito: dispiace sapere che si ossiderà molto rapidamente. Conviene dunque procedere il più presto possibile con le operazioni seguenti in modo da saldare su rame non ossidato, così da usare meno flussante e ottenere saldature migliori (gli ossidi sono isolanti elettrici).

Foratura e taglio a misura.

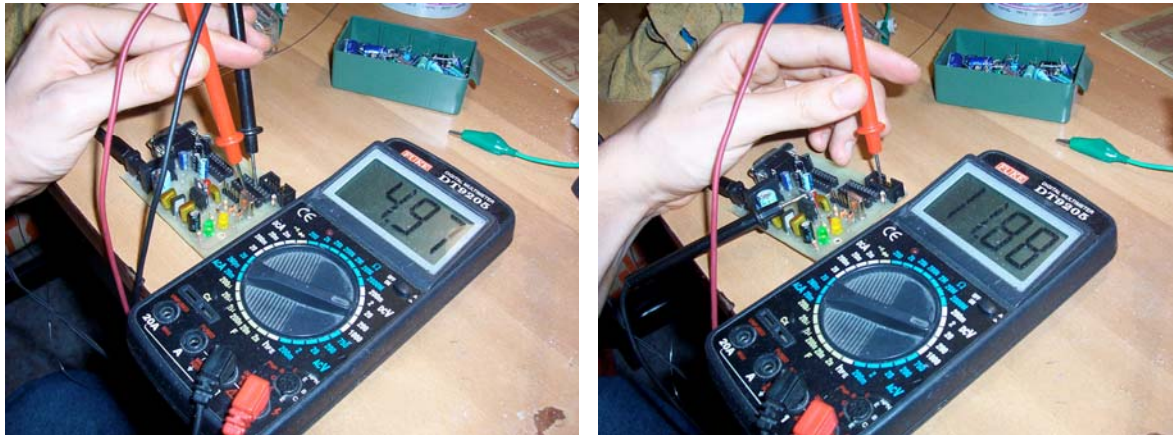
Ogni piazzola va forata con un trapano a colonna, una macchina a controllo numerico o, in mancanza di queste, con un po’ di attenzione e un comune trapanino manuale. Lo script drill-aid facilita questo compito con forellini di centratura per ogni piazzola che guidano la punta in posizione. Osservando di tanto in tanto la basetta sullo schermo del PC, si fanno i fori del diametro corretto: 0,8 mm la maggioranza, 1,0 mm per i componenti con reofori più grossi (header dei connettori, regolatori di tensione, ponte a diodi), 3,0 mm per i fori di fissaggio. In caso di dubbio sulle dimensioni di un foro si può usare il calibro sul reoforo del componente. Per qualche rara fessura di forma non circolare (lamelle del connettore di alimentazione) si può usare una mola.

Saldatura componenti

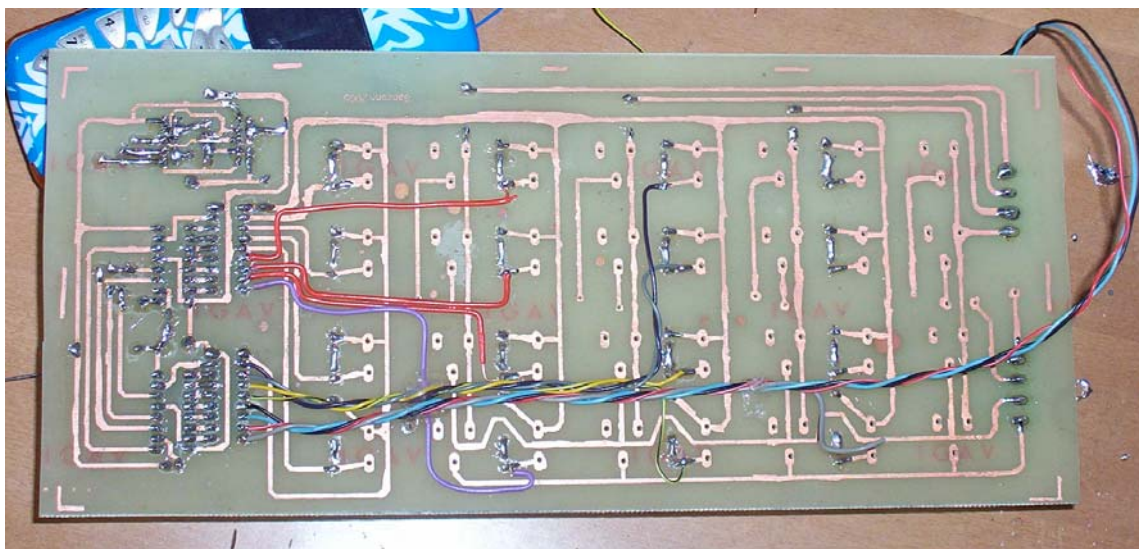
Prima di saldare i componenti è bene testare la continuità delle piste con un multimetro dotato della funzione sonora di ricerca continuità: meglio farlo adesso, che dopo aver saldato i componenti, perché ora la basetta può essere appoggiata più agevolmente sul piano di lavoro. Segnare con un pennarello le piste interrotte: occorrerà fare dei ponticelli per porvi rimedio. Verificare pure che non ci siano piste in corto, collegate tra loro per una incisione non efficace: occorre separarle con una mola e tanta attenzione.

Verifica continuità e corti piste. Saldare prima i componenti a profilo più basso.

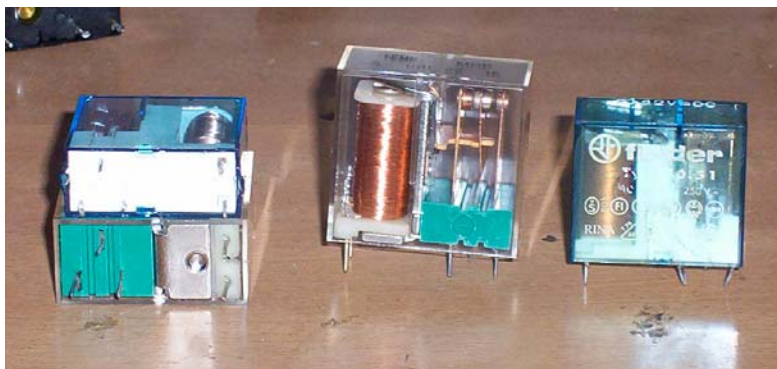
Per la saldatura conviene procedere in ordine: prima i componenti a profilo più basso, poi man mano quelli più grossi. Il motivo è ovvio: procedendo in ordine inverso si è obbligati ad usare delle pinzette per accedere al fondo della basetta, mentre con l'ordine consigliato sono sufficienti le dita di una mano. Il connettore seriale è il componente soggetto a sforzi meccanici sulla basetta. E' opportuno che durante l'uso della scheda, per le operazioni di inserimento e disinserimento del connettore, si tenga fermo direttamente il connettore e non la basetta, per evitare sollecitazioni sulle piste sottostanti, che potrebbero rompere le saldature.

Una "terza mano" aiuta a reggere la basetta. Circuito alimentato senza gli integrati inseriti.

Una "terza mano", cioè un supporto con una base pesante e due pinze orientabili, facilita le operazioni. Dopo la saldatura occorre ricontrollare continuità e corti con un multimetro, poiché una goccia di stagno può aver messo in corto due piste vicine, e l'inserimento di un reoforo può aver fatto saltare una pista.

Verifica delle tensioni nei vari punti del circuito 5V, 12V, 10V, -10V

La voglia di completare il circuito e dargli alimentazione è tanta, ma per evitare di trasformare il circuito in una nuvola di fumo bianco e componenti bruciati, ancor prima di utilizzarlo per la prima volta, è bene essere il più prudenti possibile. Dopo aver verificato continuità e corti, si può dare alimentazione al circuito, senza però inserire gli integrati. Conviene verificare con un multimetro che agli integrati arrivino tensioni di alimentazione corrette, sui piedini giusti, e con la polarità corretta. Si possono anche testare le funzionalità dell’hardware collegato agli integrati ponticellando qua e là come farebbe il microcontrollore. Questo circuito aveva molte alimentazioni da verificare: 5V, 12V, 10V, -10V. Durante le verifiche si è prestata molta attenzione a non mettere in corto, inavvertitamente, coi puntali del multimetro, due piste diverse.

La basetta con il modulo di espansione per relais è stata realizzata seguendo gli stessi principi, si evita quindi una descrizione dettagliata.

Fili rigidi con curve ad angolo retto e trecce di fili sottili fissate con colla termica.

Un elemento di novità degno di considerazione sono i 12 fili volanti da montare sul lato piste. I fili creano disordine (dannoso non solo dal punto di vista estetico, ma anche dal punto di vista funzionale, poiché rende difficile controllare che sulla basetta sia tutto a posto). Sono state seguite due filosofie diverse per realizzare cablaggi ordinati: quelli rossi e viola, in figura, sono stati realizzati con filo di rame rigido, e con curve ad angolo retto. L’idea è che curve ad angolo retto sono gradevoli agli occhi e più facili da seguire con lo sguardo rispetto a linee oblique. Però i fili occupano tanto spazio sulla basetta. I restanti cablaggi sono stati effettuati intrecciando tra loro fili non rigidi, in modo da occupare meno spazio. Le trecce sono state fissate con una goccia di colla termica. Entrambi gli approcci sono validi, e hanno consentito di lasciare libere le piazzole su cui andavano montati i relais, che inizialmente non sono stati montati, per motivi che illustrerò in seguito.

Relais in offerta speciale. Dimensioni appena differenti tra i relais acquistati in offerta e quelli tipici.

I relais, acquistati al dettaglio, costano 2,00 o 2,50 euro. Si è preferito attendere la successiva fiera dell’elettronica per cercare di acquistarne una certa quantità in offerta. Si è stati fortunati

poiché alla fiera organizzata dall'Associazione Radioamatori Italiani di Castellana Grotte, che si è tenuta a Monopoli (BA) nel mese di maggio, si sono potuti acquistare 20 relais per 10 euro.

I relais erano nuovi (cioè mai utilizzati), anche se invecchiati dagli anni, e sembravano delle dimensioni corrette per la basetta realizzata. Solo arrivati a casa ci si è resi conto che i piedini di alimentazione erano un decimo di pollice più distanti dagli altri rispetto ai relais più diffusi. I relais erano anche un po' più alti. Fortunatamente si è riusciti a montarli lo stesso piegando appena verso l'interno i piedini, ottenendo un risparmio di 30 euro e 4 relais in più di scorta, identici a quelli montati.

"Debug" hardware

Debug è di solito un termine riferito al software, e indica le noiose operazioni di aggiustamento di errori nei programmi che si aveva l'ambizione di aver progettato e scritto correttamente. Ho usato il termine debug hardware per indicare i piccoli aggiustamenti da realizzare sulla basetta per piccoli errori di progetto che è possibile risolvere con tecniche *cut & wire* o altre da valutare caso per caso. Le possibilità di intervento sono minime, quindi nel progetto di un circuito occorre dedicare molta più attenzione rispetto al progetto di un software.

ICSP non funziona

La prima brutta notizia relativa al circuito è stata che non si riusciva a programmare il microcontrollore in-circuit, ma occorreva rimuoverlo e inserirlo nel programmatore. La cosa non era tollerabile perché durante il debug del firmware occorre aggiornare continuamente la programmazione e verificare le funzionalità: i reofori del componente si sarebbero rotti dopo ripetuti inserimenti e disinserimenti nel circuito e nel programmatore.

Il problema, comune a molti circuiti, è che il programmatore di fatto si trova ad alimentare, con la sua alimentazione a 5V, tutto il circuito ospite. Se la potenza elettrica resa disponibile dal programmatore non è sufficiente, il chip non verrà programmato. Inoltre si rischiava di rompere il 7805, poiché gli si collegava alimentazione a valle (dal programmatore) senza alimentazione a monte. Una soluzione funzionante è stata interrompere, tagliandola con una fresa, una pista di alimentazione da 5V che portava corrente al microcontrollore, sostituendola con un header su cui montare un ponticello, del tipo di quelli usati nei PC per configurare le schede di espansione e la *mother board*, da inserire per l'uso abituale, e rimuovere durante la riprogrammazione. Non è la soluzione ideale, ma è comunque molto più comodo inserire e

disinserire un ponticello su un header apposito, che non un intero chip dal suo zoccolino, rischiando di romperne i pin ogni volta. Il ponticello inserito è visibile nelle foto del controller tra il led giallo e il quarzo. La correzione non è stata riportata sullo schema elettrico, poiché tale modifica non dovrebbe essere necessaria se si utilizza un programmatore che eroghi più corrente sull'uscita a 5V.

RA4 open collector

La seconda brutta notizia è stata l'aver dimenticato che la porta RA4 del microcontrollore è l'unico stadio di uscita *open drain* (tutti gli altri sono push-pull) del microcontrollore utilizzato. La porta era destinata a pilotare il segnale di clock degli shift register. Occorreva montare una resistenza di pull-up; si è preferito invece cambiare piedino di uscita, come già discusso descrivendo lo schema elettrico.

Durante il "debug hardware" di questo errore, che non si è mostrato subito e dava malfunzionamenti sporadici[45] (i più difficili da scovare), sono state collegate sonde di oscilloscopio ai vari segnali presenti sulla scheda, per capire perché gli shift-register operavano correttamente una volta si e due no. Il debug di un firmware è molto differente dal debug di un software sul PC, perché si riescono a ricavare poche informazioni sullo stato della macchina, e le variabili in gioco, che possono causare malfunzionamenti, sono numerose. Con un circuito si opera molto spesso "alla cieca" e occorre mantenere calma e concentrazione e procedere in rassegna tutte le possibili fonti di errore.

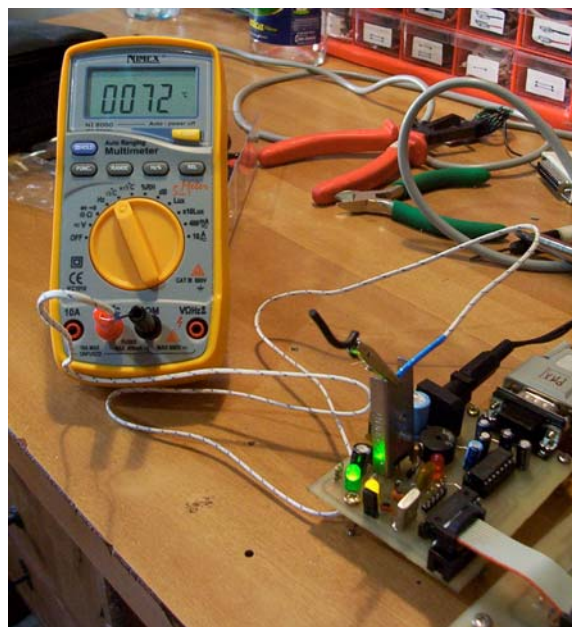
Volendo collegare sonde di oscilloscopio per misurare segnali, l'ordine corretto delle operazioni è: togliere alimentazione al circuito, saldare uno spezzone di filo alla pista da controllare, verificare col multimetro che non si siano procurati corti con l'operazione, collegare la sonda dall'altro lato del filo, alimentare il circuito. La tentazione di poggiare la sonda un po' qua un po' la, mentre il circuito è in funzione, è però grossa, e infatti ho ceduto a questa tentazione rompendo due stadi di uscita del PIC: mentre osservavo lo schermo dell'oscilloscopio ho perso di vista il puntale della sonda che ha messo in corto due uscite che il PIC teneva a livello differente. E' stato quindi necessario comprare un altro microcontrollore. Quello vecchio non è stato buttato via, perché sembra funzionare tutto

[45] A volte il circuito funzionava correttamente: il segnale di clock passava dallo stato di collegamento a massa allo stato floating, che a volte veniva letto correttamente come "1" dagli shift-register, per esempio se la pista data, adiacente, conteneva anch'essa un "1". Si ottenevano quindi malfunzionamenti molto strani e non ripetibili, che hanno finalmente orientato verso la ricerca di una linea di dati rimasta flottante senza controllo.

tranne i due stadi di uscita rotti: può essere utilizzato in un altro progetto in cui quei due pin vengano usati come ingressi o non usati affatto.

Tanto è stato il tempo necessario per scoprire questo errore, poiché tutto si pensava, tranne che una delle uscite del PIC fosse open drain (altri PIC, con la stessa piedinatura, hanno tutti i driver di uscita del tipo push-pull), che ho preso un piccolo accorgimento che evitasse l'errore in futuro. Ho modificato il componente nella libreria di Eagle, aggiungendo un asterisco nel simbolo dello schema elettrico, sul pin con stadio di uscita differente.

Dissipatore

Durante il disegno della basetta, non è stato lasciato molto spazio per il dissipatore del 7812, si è quindi dovuto spostare il foro di fissaggio ad un dissipatore piccolino a montaggio verticale, per poterlo utilizzare. Il foro è stato filettato in modo da poter fissare il dissipatore semplicemente con una vite anziché con bullone e dado, garantendo al contempo la possibilità di stringerlo più saldamente, non dovendo tener fermo il dado dall'altro lato con una pinzetta.

Foratura e filettatura del dissipatore. Dissipatore montato e test di riscaldamento con tutti i relais accesi.

Ho già discusso della potenza di alimentazione, indicando i motivi che hanno portato ad uno stadio sottodimensionato per il caso peggiore, ovvero tutti i relais accesi (condizione d'uso senza alcun interesse pratico), ma perfettamente idoneo per l'uso tipico: tra 4 e 8 relais in funzione. Il motivo principale è stato risparmiare i soldi per acquistare un trasformatore, ma utilizzare quello già disponibile. Il motivo secondario, non meno importante, esercitarsi nella

progettazione di caso tipico, anziché di caso peggiore, per contenere i costi, a discapito dell'affidabilità e della durata[46].

Volendo fare le cose meglio, è necessario sostituire il 7812 con un regolatore da 3A, e calcolare la potenza elettrica dissipata come caduta di tensione ingresso-uscita per corrente erogata. Occorre poi scegliere un dissipatore che consenta di dissipare tale potenza raggiungendo la differenza di temperatura desiderata. Le formule sono banali e si possono calcolare anche a mente, ma il suggerimento migliore è quello di sostituire semplicemente il dissipatore con uno più grosso se scalda troppo, poiché esistono molte variabili non note a priori, come il coefficiente di convezione naturale, influenzato dai componenti circostanti e dal contenitore in cui viene montato il circuito, che non è possibile valutare analiticamente, ma solo sul prototipo.

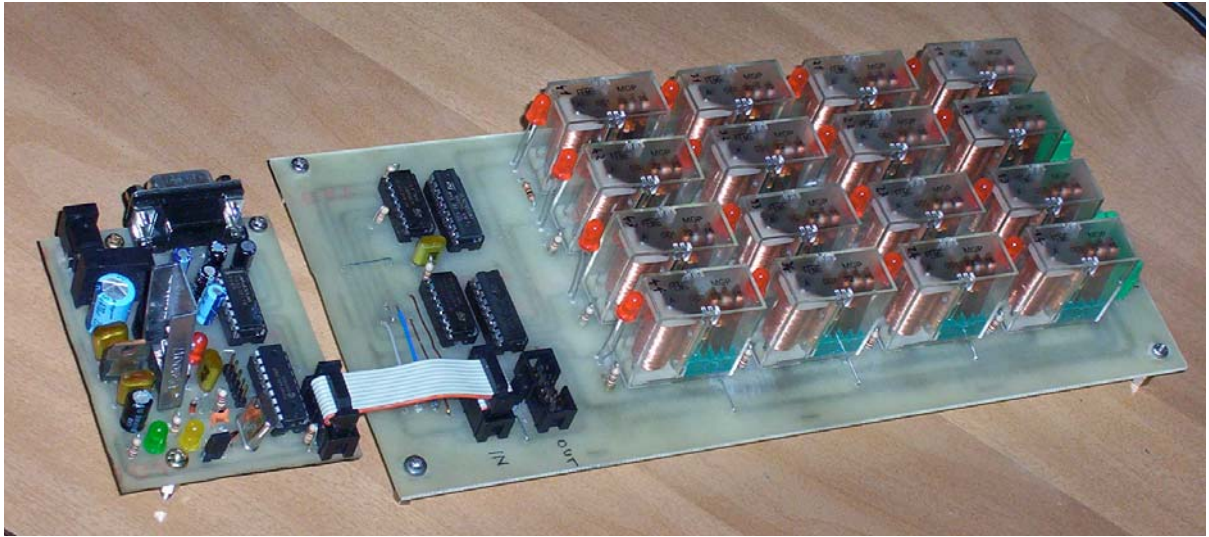
Per rendersi conto se il dissipatore scelto poteva essere adatto, ho fatto un test di riscaldamento senza troppe pretese di accuratezza. Con una temperatura ambiente di 22°C, la temperatura del dissipatore, proprio vicino al componente, a regime, con tutti e 16 i relais accesi, è stata di 84°C, un po' alta, ma comunque consentita. Il regolatore, dotato di protezione termica, non è andato in protezione. Lasciando il componente senza aletta di raffreddamento, si è potuto notare che la protezione (limitazione della corrente erogata) entrava in funzione per una temperatura esterna di 110°C (e una temperatura della giunzione presumibile di 130° circa). Con 8 relais accesi, e dissipatore montato, la temperatura a regime è stata di 62°C, con 4 relais accesi 37°C. Senza relais accesi 28°C. Per l'uso tipico (tra 4 e 8 relais accesi per volta) si attende quindi una temperatura del dissipatore di 40-60°C, perfettamente idonea.

Nessun problema di reset spuri

Nessun problema si è notato per la sezione logica facendo commutare anche molti relais tutti assieme: il microcontrollore è su una basetta separata, ha una alimentazione separata (anche se con la massa in comune), sono state bypassate le alimentazioni vicino a ciascun integrato, ma soprattutto ci sono i diodi di ricircolo vicino ai relais, e le piste di massa sono state dimensionate correttamente. Se il circuito viene progettato male, e l'alimentazione di logica e relais è la stessa, può succedere che all'atto della chiusura di un relais, la caduta di tensione sulla linea di alimentazione faccia spegnere e riaccendere il microcontrollore, che perde così tutte le configurazioni e si resetta rieseguendo il programma dall'inizio.

---

[46] La prassi è più che comune: praticamente tutti gli alimentatori per PC ATX montano componenti dimensionati per metà della potenza di targa, e infatti è esperienza comune che

Assorbimento, ingombri, pesi

Il circuito finito. E’ collegato un solo modulo, il cui terminatore è inserito.

Finito il circuito, sono state effettuate misure di assorbimento, dimensioni, peso sui circuiti, in modo da compilare il foglio delle caratteristiche tecniche elettriche e meccaniche salienti della macchina costruita.

Alimentazione	**Assorbimento**
9÷15V AC	A riposo: 30 mA
12÷18V DC (qualunque polarità)	Ogni coppia Relais + Led inserita: +110 mA
Connettore Standard 2,5x5,5x9,5 mm	Max con 1 modulo (16 relais accesi): 1,8 A
Ingombri	**Pesi**
Controller: 90x65x50 mm	Controller: 89 g
Modulo Relais: 250x114x50 mm	Modulo relais: 510 g
Wall cube: 75x62x(50+38) mm	Wall cube: 480 g

Caratteristiche elettriche e meccaniche salienti del sistema

Consuntivo costi

Prima di realizzare il progetto è stato stilato un preventivo. Viene qui mostrato il solo consuntivo, poiché le differenze tra i due sono minime.

Nel consuntivo figurano i costi per la basetta del controller e del modulo a relais, non quelli del programmatore per PIC e del circuito di test del max232. I prezzi sono al dettaglio al

tali alimentatori tendono a rompersi piuttosto facilmente.

negozio di componenti elettronici, quindi purtroppo un po' cari. Si poteva risparmiare parecchio acquistando i componenti in quantità maggiori da distributori nazionali che vendono tramite Internet.

Quantità	**Descrizione**	**Costo unitario**	**Costo x quantità**
16	Relais 12V 100mA, 1S1P	N/A	€ 10,00
2	74HC595, Shift register 8 bit	€ 0,50	€ 1,00
2	ULN2803, Array di 8 transistor Darlington NPN con diodi di clamping	€ 2,00	€ 4,00
1	PIC16F628, Microcontrollore Microchip	€ 5,00	€ 5,00
1	MAX232, Adattatore di livello TTL/Rs232	€ 3,50	€ 3,50
1	7812, Regolatore di tensione	€ 0,60	€ 0,60
1	7805, Regolatore di tensione	€ 0,60	€ 0,60
1	Ponte a diodi	€ 0,50	€ 0,50
19	Led 5mm: 17 rossi, 1 verde, 1 giallo	€ 0,20	€ 3,80
1	Buzzer piezoelettrico con oscillatore integrato	€ 1,50	€ 1,50
1	Quarzo 20 MHz	€ 2,00	€ 2,00
Vari	Condensatori: elettrolitici, multistrato, ceramici; resistori ¼ W	N/A	€ 3,00
1	Connettore di alimentazione spina volante	€ 0,70	€ 0,70
1	Connettore di alimentazione presa 90° c.s.	€ 0,50	€ 0,50
1	Connettore DSUB-9 90° C.S. Maschio	€ 1,00	€ 1,00
1	Cavo di collegamento seriale null-modem	€ 3,50	€ 3,50
2	Morsetti a vite C.S. 4 posti	€ 1,00	€ 2,00
3	Header 10 pin C.S. per cavo piatto	€ 0,30	€ 0,90
1 m	Cavo piatto 10 poli	€ 0,30	€ 0,30
3	Connettori a grimpare cavo piatto 10 poli	€ 0,30	€ 0,90
1	Dissipatore per TO220 verticale	€ 1,00	€ 1,00
1	Trasformatore AC/AC adapter	N/A	€ 0,00
2	Basette ramate	N/A	€ 2,00
Vari	Materiali di consumo: stagno, cloruro ferrico in grani, trielina	N/A	€ 2,00
1	Fotocopia con macchina a colori, uso pressa a caldo in copisteria	N/A	€ 1,00
		Totale	€ **51,30**

Il preventivo era sostanzialmente identico, ma raggiungeva la cifra di 90 euro circa, poiché considerava i relais al costo di 2,50 euro l'uno, per un totale di 40 euro per 16 relais, anziché i relais in offerta acquistati in fiera per 10 euro, e uno stadio di alimentazione correttamente dimensionato con trasformatore, raddrizzatore, condensatori elettrolitici di livellamento, cordone di alimentazione, isolamento di sicurezza da realizzare su una basetta più grande e con procedimenti di fotoincisione più costosi, anziché il trasformatore di recupero già disponibile in laboratorio.

L'attenzione al contenimento dei costi è stata fondamentale, rendendo il progetto realizzabile, poiché il budget a disposizione non raggiungeva i 90 euro.

Capitolo 3 – Studio del protocollo e progettazione logica

Il circuito è stato realizzato ancora prima di una analisi dettagliata del protocollo di comunicazione. Questo è stato possibile perché erano già chiari tutti gli aspetti dell'hardware della macchina. Era necessario però definire per bene il protocollo di comunicazione prima di stendere il firmware per non rischiare di dover fare il lavoro due volte.
Più precisi si fosse stati nella descrizione dei comandi da riconoscere e nelle risposte da fornire all'utente, più chiare si fossero avute le idee sulle possibilità di implementare in un certo modo o in un altro delle richieste, più facile sarebbe stato dopo stendere il firmware.
Durante la fase di progettazione, oltre alla mera stesura di elenchi di funzionalità da implementare, è stato necessario prendere delle decisioni riguardo a cosa implementare e cosa no, quanto rimanere fedeli allo standard dello strumento Agilent preso a modello, e quanto distanziarsi per rendere l'implementazione più semplice o attuabile.

Appunti in file di testo

Per ottimizzare i tempi, evitando di rivedere la stessa questione più volte, e per ordinare le idee, in modo da suddividere il problema complesso della realizzazione della macchina in problemi distinti, più facili da trattare singolarmente, era necessario prendere appunti il più possibile dettagliati[47] e precisi sui diversi aspetti del problema, da archiviare in maniera il più possibile ordinata in modo da poterli ritrovare non appena fosse necessario.
Per molte persone la scelta obbligata in questa prima fase è l'utilizzo di carta e penna e diagrammi con blocchi e frecce che rispecchino gli schemi mentali con i quali si sta approcciando il problema.
Anche se mi è costato fatica inizialmente, mi sono imposto di rinunciare all'immediatezza di carta, penna e schemi in favore di file di testo, che hanno il grosso vantaggio di poter essere archiviati più facilmente e nei quali si può riorganizzare l'ordine delle informazioni molto rapidamente.

---

[47] Più che un livello di dettaglio elevato, è importante utilizzare un livello di dettaglio uniforme per i diversi aspetti del problema, e dettagliare meglio dopo, sempre utilizzando un livello di dettaglio uniforme, man mano che ciascun aspetto prende corpo. Durante le prime fasi di progetto, quando molte scelte sono ancora rivedibili, è controproducente dettagliare troppo perché altrimenti occorre buttare via più lavoro in caso di revisioni parziali.

Inoltre finito il lavoro, i file di testo di appunti, utilizzati come riferimento continuo durante la scrittura del firmware, si trasformano con poche modifiche nella relazione sul lavoro e nei manuali utente del progetto realizzato, con la certezza che la documentazione sia di ottima qualità e descrizione molto fedele del progetto, in quanto nata dal cuore stesso della progettazione.
Inoltre costringere l'idea mentale a trasformarsi in periodo di testo anziché in schema rapido, aiuta a formulare correttamente tutti gli aspetti del problema.
La descrizione della progettazione logica del firmware, e del circuito in generale, procede dunque per "file di testo" successivi, ciascuno dedicato ad un aspetto del problema. Tali files costituiscono la documentazione interna del progetto, in contrapposizione alla documentazione esterna, costituita da questa relazione e il manuale per l'utilizzatore.

Glossario dei termini

Il glossario dei termini serve ad evitare il pericoloso uso di omonimi e sinonimi nella scrittura degli altri appunti, e nella documentazione finale di supporto da fornire all'utente. Inoltre contiene una descrizione precisa di cosa si intende con ciascun termine, in modo da chiarire il concetto a sé stessi, in fase di progettazione, nella documentazione interna, e all'utente finale, nel manuale da consegnare assieme all'apparecchio, o nella relazione sul lavoro svolto.
L'uso di omonimi è pericoloso perché rende poco chiari e ambigui i concetti espressi. L'uso di sinonimi è pericoloso perché può indicare che due concetti che sembrano lo stesso possono in realtà avere sfumature diverse ed dover essere implementati con due approcci differenti.
Avevo chiamato Nomenclatura unica il file di testo, e lo riporto integralmente senza modifiche per dare un'idea di cosa si tratti. Non tutto quanto descritto sarà immediatamente chiaro, ma molti concetti verranno chiariti in seguito.

Nomenclatura unica.txt

La scheda con stabilizzatore di tensione, microcontrollore, adattatore di livello si chiama Matrix Controller, o solo ***Controller***. Le schede di espansione, che ospitano i relais e i morsetti a vite facenti capo a righe e colonne si chiamano ***moduli***.
L'intero apparecchio/progetto si chiama Relais Matrix, o solo ***Matrix***.
I moduli possono essere alloggiate in 4 ***slot***, chiamati slot 100 200 300 400.
Non esistono 4 connettori per altrettanti slot, come sull'Agilent 34970A, ma i moduli vanno collegati ***in cascata***, inserendo il ***terminatore*** sull'ultimo, e verranno numerati in ordine 100 200 300 400 in base all'ordine con il quale sono collegati nella catena.
Ciascun relais ha un numero di ***canale*** associato. I numeri di canale sono numeri a due cifre in cui la prima cifra è la riga, la seconda

la colonna. L'intero numero viene però interpretato come un numero decimale ed è possibile specificare intervalli.
Le memorie in cui salvare/richiamare la combinazione attuale di relais chiusi/aperti vengono chiamate ***memorie*** (mentre vengono chiamati stati sul datasheet Agilent). Le memorie non verranno chiamate né slot, né stati, per non confondersi con gli slot in cui inserire moduli, con gli stati definiti dal protocollo IEEE-488.2 (che non ho implementato), con lo ***stato*** della macchina a stati del ***parser***, e con gli stati accessori del parser, che chiamerò ***variabili di stato***.
Header indica i connettori hardware rapidi con più pin affiancati, spaziati 1/10 di pollice. Il data sheet Agilent usa il termine Header ad indicare gli ***identificatori*** dei comandi validi. Cercherò di usare il più possibile identificatore per descrivere questi ultimi, ma a volte sarò costretto a usare header come il data-sheet Agilent.

Come si vede c'era rischio di confusione tra termini: per esempio slot, stato, memoria, potevano riferirsi tutti allo stesso concetto, ovvero le posizioni di memoria su cui salvare e richiamare lo stato attuale dei relais con i comandi *SAV e *RCL. Nella frase precedente ho volutamente usato il termine stato per indicare se un relais è acceso o spento: questo procura confusione, meglio sarebbe stato usare "condizione acceso/spento" dei relais. Questo per mostrare quanto facilmente si possa cadere in omonimie o sinonimi e quanto possa essere utile un glossario dei termini da definire il più presto possibile nel proprio lavoro.
Nel file di testo nomenclatura unica si è evitato di cadere in queste trappole della lingua naturale, cercando al contrario di essere sintetici e chiari, a scapito dell'eleganza formale. I termini chiave sono riportati ***in corsivo grassetto*** nel paragrafo in cui vengono definiti. Le definizioni sono date utilizzando il termine nel contesto, poiché è la maniera più rapida e chiara per definire termini ambigui senza impelagarsi in definizioni che utilizzino altri termini altrettanto ambigui. Preferisco questa soluzione al glossario dei termini classico, con un termine sotto l'altro con la definizione accanto, che obbliga a scorrere avanti e indietro nel dizionario per cercare di orientarsi tra i continui richiami.

Appunti per la relazione

Durante il procedere del lavoro, si notavano aspetti particolari e soluzioni più o meno brillanti dei piccoli problemi incontrati, che era un peccato non mettere nella relazione finale. Era stata richiesta inoltre dal professore una documentazione dettagliata sull'intero progetto, ed è facile che risolto un problema, ci si dimentichi della strada percorsa, attraverso soluzioni parziali poi scartate, per arrivare alla soluzione finale, ricordando alla fine non già l'intero percorso, ma solo i punti di partenza e di arrivo, vale a dire il problema iniziale e la soluzione finale.
Ho allora appuntato in un file di testo, alla rinfusa, i concetti da inserire nella relazione, ottenendo la tranquillità di appuntare tutte le idee man mano che venivano in mente, senza

curarmi subito se, dove e come era giusto inserire ciascuna idea nella relazione: era importante, nella fase iniziale, non perderne neanche una. La stesura della relazione è consistita solo nel riordinare queste idee, suddividendole in capitoli e paragrafi opportuni, espandendo i contenuti là dove serviva, corredando di foto e diagrammi dove opportuno.

Scheda video dual-head con supporto per due monitor, su cui sono aperti documenti differenti.

Ogni volta che era necessario modificare un file di testo, riorganizzando l’ordine dei concetti, e per la stesura della relazione, mi sono avvalso della scheda video dual-head, vale a dire con la possibilità di generare due immagini differenti su due monitor. Le operazioni di taglia/incolla di porzioni di testo da un documento ad un altro erano così immediate e si potevano tenere sott’occhio contemporaneamente gli appunti e il testo definitivo.

Decisioni da prendere, decisioni prese

Durante la definizione dei diversi aspetti del problema, capita continuamente di dover prendere decisioni su come procedere ed effettuare scelte più o meno definitive e più o meno rivedibili. Spesso diverse scelte sono collegate tra loro, e a volte non è possibile prendere una decisione su un aspetto del problema se non si dettaglia meglio un altro aspetto del problema separato concettualmente ma intimamente legato nell’implementazione o viceversa.

Altre volte alcune decisioni facilitano un compito ma ne rendono più difficile un altro, e occorre cercare un compromesso, oppure scegliere la via che globalmente rende il progetto più facile, anche a scapito di rendere più difficile un piccolo particolare, o soddisfa globalmente più richieste del cliente, se non è possibile soddisfarle tutte.

Capita anche che sia indifferente implementare una specifica in un modo o in un altro, e si desidera appuntare entrambe le possibilità, anziché sceglierne da subito una, poiché altri aspetti del problema possono indicare che una via è da preferire perché facilita altre questioni connesse.

Cercare di tenere tutte queste questioni a mente per cercare il compromesso migliore è impossibile, allora ho appuntato le decisioni da prendere, le differenti alternative e le decisioni già prese, indicando quali erano definitive e quali rivedibili, in un apposito file di testo a più sezioni.

Inizialmente la sezione "decisioni da prendere/aspetti da fissare" era la più corposa. Man mano si è però sfoltita, trasferendo i suoi contenuti in "decisioni prese/scelte effettuate". Ogni decisione presa indicava poi la strada per le altre collegate. Ogni decisione da prendere poteva indicare la necessità di sperimentare ancora con il circuito o di documentarsi meglio per risolvere il dilemma.

Una volta svuotata la sezione "aspetti da fissare", questa si riempiva da capo quando si desiderava dettagliare meglio qualche aspetto, facendo esplodere differenti scelte implementative di livello più basso. Questa fase di organizzazione delle idee, prima, e dei dettagli implementativi spiccioli dopo, è essenziale e determinante per la buona riuscita: più tempo si dedica alla progettazione, più diretta sarà la scrittura del codice. Le revisioni, quando ci si accorge di una scelta infelice, sono possibili, ma pesanti, frustranti (perché spesso si butta via molto del lavoro fatto) e fonte di errore se non si aggiorna contemporaneamente tutto quanto si è implementato e documentato con la modifica proposta.

Questa sezione di appunti è informe, in continuo divenire, e cessa di esistere quando il progetto è finito poiché tutte le scelte sono state effettuate, e riportate nella documentazione delle specifiche, e tutti i dettagli implementativi realizzati, divenuti ormai circuito e firmware.

Mostro però alcuni esempi di questi appunti per dare un'idea più concreta di che si tratta, e per rendere più snella la successiva descrizione del firmware. In questo paragrafo descrivo le scelte che hanno portato ad una certa implementazione, nel paragrafo di commento al firmware descrivo come i costrutti assembler utilizzati implementano queste idee, senza ripetere le considerazioni di principio e di opportunità, fornite qui.

```
Il parser sarà implementato con una macchina a stati. Lo stato
principale verrà indicato dal punto di esecuzione corrente. Altre
variabili di appoggio memorizzeranno eventuali costrutti già
riconosciuti. Non ci sarà la variabile di stato principale, ma
direttamente la posizione di esecuzione indicherà in che stato ci si
trova. Se si riesce, ed è una idea valida, si userà lo stack
hardware del PIC come stack degli stati, evitando di implementarne
uno software.
```

Queste erano decisioni iniziali per il parser che sono state poi riviste, modificate e stravolte, ma sono servite come punto di partenza e paletto fissato per costruire il resto del progetto, senza pensare più al parser. Aver fissato le idee sul parser ha consentito di analizzare la sintassi dei comandi da riconoscere, verificando che il parser, così come era stato pensato, sarebbe stato in grado di farlo. Finito di definire l'elenco delle sintassi che il parser doveva riconoscere, si è dovuta rivedere la struttura del parser. Il parser finale ha utilizzato una variabile per indicare lo stato principale, anziché il punto di esecuzione, non ha usato uno stack di stati, né hardware, né software, sopperendo alla sua mancanza con molte variabili di stato ausiliarie.

Ecco un esempio di decisioni obbligate dettate dal data-sheet dello strumento Agilent che descrive la sintassi da riconoscere. Pur essendo tutto fissato, c'è ancora qualche grado di libertà possibile.

Channel List Parameters Channel list parameters specify one or more channel numbers on the module in the specified slot. The channel list must be preceded with the " @ " symbol and enclosed in parentheses. The following command uses a *channel list* parameter:

```
ROUTe:CHANnel:DELay? (@<ch_list>)
```

- The following command configures a scan list to include only channel 10 on the module in slot 300.

```
ROUT:SCAN (@310)
```

- The following command configures a scan list to include multiple channels on the module in slot 200. The scan list now contains only channels 10, 12, and 15 (the scan list is redefined each time you send a new ROUTe:SCAN command).

```
ROUT:SCAN (@210,212,215)
```

La pagina del manuale Agilent che descrive la sintassi per le liste di canali

Ed ecco gli appunti presi a riguardo:

Le liste di canali hanno questa sintassi:
(@101)
(@101,102,204)
(@101:104,106)
Gli estremi devono essere validi col circuito attuale. Negli intervalli con i due punti si possono sorvolare posizioni non valide, ma inizio e fine devono essere validi. Man mano che si riconoscono numeri di slot, viene preparata in memoria la lista degli slot che l'operazione corrente interesserà. Se il comando termina correttamente, la lista verrà utilizzata, altrimenti verrà scartata. Non si supporta la sintassi coi :, però si segnala errore non implementata, non errore generico di sintassi non riconosciuta, così che in revisioni successive del firmware la si possa implementare, scrivendo il codice nel posto lasciato libero (*place holder*) nel codice.

Un vincolo di sintassi, e la necessità di segnalare errori, ha indicato che era necessario poter ritardare l'esecuzione del comando fino al raggiungimento della fine della riga corrente, ed era quindi necessaria una struttura di memoria ausiliaria che memorizzasse la lista di canali da aggiornare, anziché aggiornare ogni relais man mano che si riconosce un numero di canale valido. La lunghezza della lista può essere anche molto lunga, e non c'è nel PIC la memoria ram a disposizione per memorizzare un'intera riga, cercare errori di sintassi, e processarla nuovamente, eseguendola man mano, se non sono stati trovati errori. E' stato quindi necessario cercare soluzioni alternative, come la lista di canali memorizzata internamente in maniera molto compatta. Questo è un esempio di come una specifica (la sintassi delle liste di canali) può interessare profondamente differenti aspetti, anche importanti, del problema, che non si ritenevano inizialmente collegati tra loro (la gestione del flusso e l'assenza di un buffer di ricezione): in tali casi è una benedizione aver fatto la scelta di dettagliare il più possibile contemporaneamente le diverse questioni, a caccia delle interrelazioni possibili.

Ecco alcune decisioni spicciole su cui c'era massima libertà di scelta, ma che andavano comunque prese.

```
La porta RS232 verrà utilizzata con nessun controllo di flusso. 57600 baud, 8 bit, nessuna parità, 1 bit di stop.

Lo strumento Agilent ha un comando che richiede di memorizzare continuamente, in memoria non volatile, il default di power-on e di ripristinarlo alla riaccensione. Non implemento questa cosa.

Lo strumento deve fare beep quando si verifica un errore.

Lo strumento accetterà i due punti a inizio riga, che possono anche essere omessi, come prescrive lo standard.

Si onorerà il discorso dei sub-system e dell'ultimo sub-system usato, non si onorerà il ; per inviare più comandi di seguito. Questo significa che si memorizza l'ultimo sub-system usato in una variabile di stato accessoria.

Non conterò il numero di azionamenti di ciascun relais, a prevedere l'istante di rottura dalla vita media, perché non c'è abbastanza spazio nella eeprom del microcontrollore, che verrà utilizzata per salvare/richiamare configurazioni di relais, che giudico uno scopo più utile.

Il led rosso indica che si è verificato un errore, e si attende solo il termine della riga corrente, senza processare più nulla. L'errore verrà comunicato al termine della riga, per evitare di perdere caratteri in ricezione e generare un buffer over-run.

Tutti i comandi sono case-insensitive.
```

La questione del fine-riga, che può apparire banale, è in realtà molto delicata.

> Come separatore di riga verrà riconosciuto new-line 0x10. I caratteri 0x0D verranno semplicemente scartati, così che possa essere riconosciuta anche la combinazione 0x0D 0x10, anche se non viene controllato che i caratteri siano consecutivi. Le stringhe non verranno prima bufferate fino a 0x10, e poi interpretate, ma direttamente interpretate man mano che arrivano, poiché alcuni comandi possono essere molto lunghi e non c’è abbastanza memoria nel PIC per conservarli. L’operazione interpretata, riconosciuta e memorizzata nelle variabili di stato ausiliarie, non verrà però eseguita prima del raggiungimento del fine-riga, per lasciare comunque la possibilità di disfare un comando che sembrava con sintassi valida ma contiene un errore proprio sul finale.

Ed ecco una decisione che ha ridotto l’occupazione in memoria del firmware e facilitato enormemente la scrittura del parser.

> I comandi possono essere abbreviati alla prima abbreviazione che li distingue uno dall’altro, e dopo di che possono continuare come capita, verranno sempre riconosciuti come validi.

Quindi ROUT, ROUTE, ROUTContinuoComeMiPare verranno riconosciuti sempre come lo stesso comando valido. Così facendo è necessario memorizzare solo la parte obbligatoria degli identificatori nel programma, anziché l’identificatore per intero, risparmiando memoria e facilitando il codice. E infine una decisione i cui risvolti sono chiariti nei prossimi paragrafi.

> Dopo ogni comando ricevuto verrà annunciato che è stato eseguito, oppure che c’è stato un errore. Non si gestisce nessuna queue di errori o bit di stato dello strumento. La compatibilità con lo standard IEEE488.2 sarà a livello di sintassi, non di architettura interna, né di codici di errore.
> I messaggi avranno tutti la stessa struttura: codice numerico e stringa descrittiva. Il codice numerico sarà espresso in decimale partendo da dati a 8 bit, perché il microcontrollore usato è a 8 bit, e i messaggi di errore non sono tanti di numero.

SCPI (IEEE-488.2)

Questo è il nome dello standard industriale rispettato da moltissima strumentazione di misura da laboratorio o industriale. Si aveva inizialmente l’ambizioso desiderio di realizzare una macchina perfettamente compatibile con lo standard. Si sono pertanto studiati il manuale del programmatore e il manuale utente dell’unità di acquisizione dati 34970A e della matrice di relais 4x8 34904A, entrambe della Agilent. Si sono presi appunti su come funziona lo standard e si è stilata una lista dei comandi da implementare e di quelli da non implementare nella propria macchina. Segue una traduzione leggibile di questi appunti.

SCPI sta per *Standard Commands for Programmable Instruments*[48], ed è un linguaggio basato su *ASCII*[49] sviluppato appositamente per strumenti di misura e di test. I comandi disponibili

[48] Traduzione letterale: Comandi standardizzati per strumentazione programmabile.

[49] ASCII sta per *American Standard Code for Information Interchange* (Traduzione letterale: Codice standardizzato Americano per l’interscambio di informazione), ed è nato negli anni

sono organizzati in una struttura gerarchica ad albero, in cui i diversi rami, principali e secondari, vengono chiamati subsystems. Anche i nomi dei subsystem sono stati standardizzati e strumenti con funzionalità simili, di produttori diversi, che rispettano lo stesso standard, implementano gli stessi subsystem dello standard, anche se possono differenziare la sintassi dei singoli comandi. IEEE-488.2 è il nome dello standard internazionale che descrive SCPI.

Lo standard IEEE-488.2 definisce un insieme di *common commands* per funzioni come il reset delle impostazioni di fabbrica, self-test e indicazioni di stato. I common commands iniziano sempre con il carattere asterisco, e proseguono con un identificatore di 3 o 4 caratteri. E' consentito usare il ; per indicare più comandi sulla stessa riga, se i differenti comandi appartengono allo stesso subsystem. Il simbolo : (due punti) separa l'identificatore del comando di un livello dall'identificatore del comando di livello inferiore. L'ultimo subsystem utilizzato diventa il subsystem di default, e comandi successivi, dello stesso subsystem, possono omettere l'indicazione iniziale del subsystem.

Lo standard non prevede i due punti all'inizio della riga, ma la maggioranza degli strumenti da laboratorio lo accetta senza obiettare. Ogni riga è terminata dal carattere 0x10 (new-line).

Si può richiedere il valore attuale della maggior parte delle impostazioni aggiungendo un punto interrogativo in coda al comando, che lo trasforma in una query.

Device clear è un messaggio di basso livello sul bus di comunicazione che riporta lo strumento in uno stato in cui risponde all'utente, abortendo operazioni in corso di durata elevata. Per gli strumenti con RS232, la ricezione di un CTRL-C sortisce lo stesso effetto. La cosa non è necessaria sul mio strumento poiché non ci sono operazioni particolari che richiedono un tempo di elaborazione lungo, e la scheda si mantiene sempre pronta a rispondere. Per compatibilità con lo standard, i CTRL-C verranno semplicemente ignorati.

L'apparecchio costruito, se non si può dire che rispetti lo standard in tutto e per tutto, utilizza comunque la stessa sintassi per quanto riguarda i comandi da riconoscere e le risposte alle query, mentre si differenzia nella gestione dei messaggi di errore, che vengono inviati subito, anziché essere accodati per una richiesta successiva.

'80 per impedire il prolificare di codici di codifica incompatibili tra differenti dispositivi per la gestione automatizzata dell'informazione.

Comandi implementati

Fase decisionale

Sono stati esaminati tutti i comandi presenti sul datasheet dell'Agilent 34970A, per vedere quali potevano avere un senso per la matrice di relais costruita, e di questi si è riportata una breve, ma il più possibile precisa descrizione del funzionamento che avrebbero dovuto avere sulla macchina. Sul data-sheet Agilent veniva infatti descritta sintassi ed effetti dei comandi, non certo i dettagli implementativi per la loro realizzazione.
In seguito si è esaminato ogni comando decidendo quali implementare e quali no, e a quali dare la priorità tra quelli da implementare, poiché il tempo a disposizione per completare il progetto non era moltissimo. Di qualche comando si è cambiata, o semplificata, la sintassi per rendere l'implementazione più semplice. Qualche comando particolarmente gravoso da implementare, ma giudicato molto utile, è stato comunque implementato con un supplemento di sforzi e impegno.

Fase finale

Questo paragrafo costituisce a tutti gli effetti la guida di riferimento rapida per il programmatore dell'apparecchio costruito, infarcito di qualche considerazione di progetto. Di ciascun comando viene descritto lo scopo, il modo di funzionare e la sintassi corretta nell'implementazione realizzata. Come termine di paragone, racchiuse in un riquadro, vengono mostrate le pagine del datasheet dell'Agilent, con la descrizione del comando originale preso a modello, per dare una idea di quanto si sia riusciti a mantenersi aderenti allo standard.

Common commands

*RST

Resetta lo strumento, così come se fosse stato spento e riacceso. Viene effettuata nuovamente la routine di autotest iniziale e la verifica del numero di moduli collegati. Se l'autotest ha successo, viene mostrato il messaggio di benvenuto *02 Hallo*, altrimenti uno dei due messaggi di errore *09 Interrupted chain* o *15 Too many modules*.

***RST**

Reset the instrument to the Factory configuration. See "Factory Reset State" on page 160 in chapter 4 for a complete listing of the instrument's Factory Reset state.

*IDN?

E’ la richiesta della stringa di identificazione dello strumento. Lo standard prevede una risposta costituita da Ditta produttrice, Modello, Numero di revisione dell’hardware, Numero di revisione del Firmware. La revisione del firmware può essere espressa con trattini per indicare più numeri di revisione. Il 34970A della Agilent distingue tre numeri di revisione per il firmware, il mio circuito indica come versione 1.0 tanto l’hardware che il firmware, rispondendo alla query con la stringa:

```
PaoloSancono,RelaisMatrix,1.0,1.0
```

***IDN?**

Read the instrument's identification string. The instrument returns three numbers for the system firmware. The first number is the firmware revision number for the measurement processor; the second is for the input/output processor; and the third is for the front-panel processor. An example string is shown below:

HEWLETT-PACKARD,34970A,X,X.X-X.X-X.X

Be sure to dimension a string variable with at least 40 characters.

*TST?

Effettua una routine di autotest, ovvero di autodiagnostica delle funzionalità hardware/software. Nel nostro caso viene inviato un pattern di bit sugli shift register verificando che ritorni, ritardato, lo stesso pattern. La risposta è la stringa di due caratteri “+1” se l’autotest ha successo, la stringa di due caratteri “+0” se l’autotest fallisce. In caso di fallimento, il controller invia un ulteriore messaggio di errore con il tipo di errore hardware rilevato: *09 Interrupted chain* o *15 Too many modules*. Dopo un autotest, viene valutato nuovamente il numero di moduli di espansione effettivamente collegate al controller, numero conservato in un registro interno interrogabile con SYSTEM:MODULE?

***TST?**

Perform a complete self-test of the instrument. Returns "+0" if the self-test is successful, or "+1" if the test fails.

*SAV {0|1|2|3|4|5}

*RCL {0|1|2|3|4|5}

Questi comandi sono descritti in seguito, nell’ambito del subsystem MEMORY.

*CLS

Questo comando non ha senso per Relais Matrix poiché non sono stati implementati i registri interni che dovrebbero essere cancellati da questo comando (flag di evento occorso, code di errori e allarmi ricevuti, tutte funzioni non implementate). Il comando viene comunque riconosciuto e processato, e viene fornita la risposta *01 Completed* anziché l'errore *03 Unimplemented* o *04 Unknown Header*.

Il codice necessario per implementare *CLS è stato codificato come *place holder*[50] per sviluppi futuri: se il comando deve eseguire una routine, basta inserirla nel corretto punto del codice, che è stato già scritto. Similmente, volendo implementare un altro dei common commands non implementati, basta duplicare la struttura implementativa di *CLS per il comando che si desidera supportare.

> ***CLS**
>
> Clear the event register in all register groups. This command also clears the error queue and the alarm queue.

*PSC {0|1}

*ESE <enable_value>

*SRE <enable_value>

*ESR?

*ESE?

*OPC

*STB?

*SRE?

*PSC?

*WAI

*TRG

Tutti questi comandi non avevano senso per il circuito Relais Matrix e non sono stati implementati. Con la revisione corrente del firmware, si ottiene in risposta ad uno di questi comandi l'errore *04 Unknown Header*. Volendo rispondere con *03 Unimplemented*, occorre semplicemente aggiungere una struttura analoga a quella presente nel codice per *CLS. Mi

[50] Trad. letterale: Segna-posto.

sembrava inutile aggiungere controlli al parser solo per mostrare un messaggio di errore differente per questi comandi rispetto all’immissione di un identificatore realmente sconosciuto.

Subsystem ROUTE

E’ il subsystem più importante, in quanto contiene i comandi per aprire e chiudere i relais presenti sui moduli di espansione.

La descrizione della sintassi dei questo comando, utilizza le lettere maiuscole per indicare la parte necessaria dell’identificatore, le minuscole per la parte facoltativa. Ricordo che nella mia implementazione, dopo la parte obbligatoria, si può completare il comando con un numero arbitrario di caratteri alfabetici senza generare errori, per i motivi discussi altrove in questa relazione.

Le liste di canali vengono indicate con (@ch_list), simbolo che va sostituito con la lista attuale di canali quando si invia il comando allo strumento. I canali sono indicati con numeri a tre cifre: la prima indica lo slot al quale è collegato il modulo di espansione. Numeri di slot validi sono 100, 200, 300, 400, con gli slot numerati in base all’ordine nel quale i moduli sono presenti nella catena di collegamento in cascata. Le altre due cifre indicano la riga e la colonna, che possono valere da 1 a 4. Una lista di canali inizia con i due caratteri “(@” (parentesi tonda aperta e chiocciola), prosegue con numeri di canale a tre cifre separati dal carattere “,” (una virgola) e termina con una parentesi chiusa. E’ possibile indicare una lista vuota con la sequenza di tre caratteri “(@)”.

Le liste dichiarate per intervallo, tramite l’uso del carattere “:” (due punti) non sono supportate dalla revisione attuale del firmware, ma il carattere di due punti viene correttamente processato dal parser e provoca l’emissione dell’errore *03 Unimplemented*, anziché il generico *05 Invalid Char*. Questo perché nel codice del firmware c’è un *place holder* per supportare in revisioni future le liste di canali dichiarate per intervalli, che non si è avuto il tempo di implementare.

ROUTe:CLOSe (@ch_list)

Chiude i relais specificati. Viene effettuata la mascheratura dei relais già chiusi, che non verranno aperti dal comando.

ROUTe:CLOSe:EXCLusive (@ch_list)

Comportamento simile a ROUTE:CLOSE: i relais specificati vengono chiusi, ma in questo caso non viene effettuata la mascheratura, quindi verranno chiusi i relais specificati e aperti tutti gli altri.

ROUTe:CLOSe? (@ch_list)

La query, in questa forma, dovrebbe restituire un “1” per ciascun relais chiuso e uno “0” per ciascun relais aperto, tra quelli indicati nella lista. Gli “1” e “0” devono essere restituiti nello stesso ordine nel quale compare la lista di canali nella richiesta. Questa query non è stata implementata in questa forma perché avrebbe richiesto un buffer in memoria di lunghezza non nota a priori per memorizzare la lista di canali, o almeno la risposta da fornire, da inviare poi al termine della riga se il comando non avesse contenuto errori di sintassi. Gestire buffer di lunghezza arbitraria in un dispositivo con poca memoria era difficile, avrebbe richiesto del tempo, e si è preferito modificare leggermente la sintassi della richiesta e della risposta per questa query, con il comando seguente, più facile da implementare, dedicando il tempo risparmiato a questioni più interessanti.

ROUTe:CLOSe?

Questo tipo di richiesta non è previsto dallo strumento Agilent di cui si vuole copiare la sintassi, ma è stato implementato in Relais Matrix con una sintassi per la richiesta e per la risposta proprietarie.

La richiesta non prevede una lista di canali, ma la semplice richiesta della condizione di apertura/chiusura di tutti i relais. La risposta prevede i primi due caratteri che indicano, mediante un numero decimale, quanti altri caratteri attendere nella risposta. I primi due caratteri possono essere 00, 16, 32, 48, 64, a seconda del numero di moduli collegati. Seguono quindi 0 e 1 ad indicare relais aperti o chiusi, ordinati dal numero di canale più piccolo al più grande. Infine il carattere new-line.

La sintassi della risposta è simile a quella di WAVEFORM:POINTS? degli oscilloscopi Agilent: i primi caratteri indicano quanti altri caratteri sono presenti nella risposta.

Switch Control Commands

ROUTe
:CLOSe (@<*ch_list*>)
:CLOSe:EXCLusive (@<*ch_list*>)
:CLOSe? (@<*ch_list*>)

Close the specified channels on a multiplexer or switch module. If any multiplexer channels are configured to be part of the scan list, you cannot close multiple channels on that module; closing one channel will open the previously closed channel. On the other modules, you can use the :EXCL command to ensure that all channels are open before closing the specified channel. The :CLOS? query returns the state of the specified channels. Returns "1" if the channel is closed or "0" if the channel is open.

- On the 20-channel multiplexer (34901A), only one of the shunt switches (channels 21 and 22) can be closed at a time; connecting one channel will close the other.
- On the matrix module (34904A), you can close multiple channels at the same time. Each crosspoint relay on this module has its own unique channel label representing the row and column. For example, channel 32 represents the crosspoint connection between row 3 and column 2.

ROUTe:OPEN (@ch_list)

Apre i relais specificati.

ROUTe:OPEN? (@ch_list)

Non implementato.

ROUTe:OPEN?

Comportamento analogo a ROUTe:CLOSe?. Uno “0” rappresenta però un relais chiuso, un “1” un relais aperto.

ROUTe
:OPEN (@<*ch_list*>)
:OPEN? (@<*ch_list*>)

Open the specified channels on a multiplexer or switch module. The :OPEN? query returns the state of the specified channel. Returns "1" if the channel is open or "0" if the channel is closed.

Subsystem SYSTEM

SYSTem:CPON {100|200|300|400|ALL}

Apre tutti i relais del modulo nello slot specificato. Specificando ALL verranno aperti i relais di tutti i moduli collegati. Lo strumento Agilent è dotato di 3 slot, mentre Relais Matrix ne supporta 4. Lo strumento Agilent supporta condizioni di accensione differenti da tutti i relais aperti, impostabili dall'utente, mentre Relais Matrix avrà sempre tutti i relais aperti all'accensione.

SYSTem:CPON {100|200|300|ALL}

Reset the module in the specified slot to its power-on state (CPON means "card power on"). To reset all three slots, specify ALL.

SYSTem:VERsion?

Viene indicata la versione del protocollo SCPI utilizzato. Relais Matrix risponde, così come lo strumento Agilent, con la stringa "1994.0" anche se non supporta pienamente lo standard.

SYSTem:VERSion?

Query the instrument to determine the present SCPI version. Returns a string in the form "YYYY.V", where "YYYY" represents the year of the version, and "V" represents a version number for that year (e.g., 1994.0).

SYSTem:MODUle?

Questa query è stata inventata apposta per Relais Matrix, e rappresenta quindi una estensione proprietaria dello standard. Viene restituito il numero di moduli trovato collegato nel corso dell'ultima accensione, reset (*RST) o autotest (*TST?) del dispositivo. La risposta è un singolo carattere, seguito dal terminatore di riga, che può essere "1" "2" "3" "4", in base al numero di moduli collegati, oppure "0" ad indicare che la catena degli shift register è stata trovata interrotta, oppure "5" ad indicare che sono stati trovati più di 4 moduli collegati.

NOTA: con il comando SYSTEM:MODULE? non viene ricontrollata la funzionalità della catena di shift-register, ma solo letto il valore memorizzato in un registro interno. Per ricontrollare la catena e aggiornare tale valore utilizzare *TST?.

Subsystem MEMORY

I comandi di questo subsystem consentono di memorizzare configurazioni di relais in 5 memorie non volatili (che ritengono l'informazione anche in assenza di alimentazione al circuito) e in una temporanea, volatile, utilizzabile finché il circuito resta alimentato.

Al momento in cui si scrive, i comandi di questo subsystem non sono stati ancora implementati, per mancanza di tempo, nel firmware, ma ne è stato specificato chiaramente il comportamento, in modo da renderne facile l'implementazione. Saranno implementati in una revisione successiva del firmware.

State Storage Commands

The instrument has six storage locations in non-volatile memory to store instrument states. The locations are numbered 0 through 5. The instrument uses location "0" to automatically hold the state of the instrument at power down. You can also assign a name to each of the locations (1 through 5) for use from the front panel.

*SAV {0|1|2|3|4|5}

Store (save) the current instrument state in the specified storage location. Any state previously stored in the same location is overwritten (no error is generated).

- You can store the instrument state in any of the six locations. However, you can only recall a state from a location that contains a previously stored state. You can use location "0" to store a sixth instrument state. However, keep in mind that location "0" is automatically overwritten when power is cycled.
- The instrument stores the state of all modules including all channel configurations, scanning setups, alarm values, and scaling values.
- A Factory Reset (*RST command) does not affect the configurations stored in memory. Once a state is stored, it remains until it is overwritten or specifically deleted.

*RCL {0|1|2|3|4|5}

Recall the instrument state stored in the specified storage location. You cannot recall the instrument state from a storage location that is empty or was deleted. When shipped from the factory, storage locations "1" through "5" are empty (location "0" has the power-on state).

- You can use location "0" to store a sixth instrument state. However, keep in mind that location "0" is automatically overwritten when power is cycled.
- Before recalling a stored state, the instrument verifies that the same module types are installed in each slot. If a different module type is installed, the instrument will perform the equivalent of a Card Reset (SYSTem:CPON command) on that slot.

*SAV {0|1|2|3|4|5}

Salva la configurazione on/off attuale dei relais in una EEPROM non volatile, assieme al numero di moduli che erano collegati alla macchina all'atto del salvataggio e un flag per indicare che la memoria di posto indicato ha un salvataggio valido. Nessun errore viene prodotto per la sovrascrittura di una posizione già occupata. La memoria 0 viene settata all'accensione con il default di power-on dello strumento (nel nostro caso, tutti relais aperti). Può essere modificata e riletta successivamente, come le altre 5 memorie, ma non conserva i dati in mancanza di alimentazione. Sulla macchina Agilent le condizioni di power-on possono essere diverse da tutti relais aperti.

*RCL {0|1|2|3|4|5}

Ripristina la configurazione on/off dei relais leggendola dalla memoria specificata. Viene prodotto un errore per il tentativo di rileggere una posizione che non conteneva dati, oppure dati relativi ad una situazione differente di moduli montati negli slot. La memoria zero sarà sempre valida perché all'accensione viene scritto il numero di moduli effettivamente collegati, dopo aver fatto l'autotest.

MEMory:STATe:DELete {0|1|2|3|4|5}

Cancella la locazione di memorizzazione. Nessun errore segnalato in caso di cancellazione di uno slot che non conteneva dati.

MEMory:STATe:DELete {0|1|2|3|4|5}

Delete the contents of the specified storage location. If you have named a storage location (MEM:STAT:NAME command), this command *does not* remove the name that you assigned. Note that you cannot recall the instrument state from a storage location that was deleted. An error is generated if you attempt to recall a deleted state.

MEMory:STATe:VALid? {0|1|2|3|4|5}

Indica se la memoria specificata è piena o vuota. Risponde con 1 o con 0.

MEMory:STATe:VALid? {0|1|2|3|4|5}

Query the specified storage location to determine if a valid state is currently stored in this location. You can use this command before sending the *RCL command to determine if a state has been previously stored in this location. Returns "0" if no state has been stored or if it has been deleted. Returns "1" if a valid state is stored in this location.

Prospetto riassuntivo

Può fare un effetto curioso, ma il prospetto riassuntivo dei comandi, che sottintende le scelte fatte su quali e quanti comandi implementare, assieme alle indicazioni sulle strategie migliori per implementarli, rappresenta più di metà dell’intero lavoro di progettazione, condensato in una tabella di una ventina di righe.

Common Commands	**SubSystem Route**
Implementati *RST *IDN? *TST?	ROUTe:CLOSe (@ch_list) ROUTe:CLOSe:EXCLusive (@ch_list) ROUTE:OPEN (@CH_LIST) ROUTe:CLOSe? ROUTe:OPEN?
Implementazione futura *SAV {0|1|2|3|4|5} *RCL {0|1|2|3|4|5}	**SubSystem System**
Place holders *CLS	SYSTem:VERsion? SYSTem:CPON {100|200|300|400|ALL} SYSTem:MODULe?
Non implementati *PSC {0|1} *PSC? *ESE <value> *ESE? *SRE <value> *SRE? *OPC *WAI *TRG *STB? *ESR?	**Subsystem Memory** *Implementazione futura* MEMory:STATe:DELete {0|1|2|3|4|5} MEMory:STATe:VALid? {0|1|2|3|4|5}

Tabella riassuntiva dei comandi disponibili

Messaggi di errore ed informativi

I messaggi di errore sono stati appuntati in un apposito file di testo man mano che, durante la progettazione, ne sorgeva la necessità. Sono stati utili come documentazione interna per lo sviluppo del firmware, e costituiscono anche la documentazione per l’utente finale che dovrà usare il circuito.

Bit Definitions – Status Byte Register

Bit Number	Decimal Value	Definition
0 *Not Used*	1	*Returns "0".*
1 Alarm Condition	2	One or more bits are set in the Alarm Register (bits must be enabled).
2 *Not Used*	4	*Returns "0".*
3 Questionable Data	8	One or more bits are set in the Questionable Data Register (bits must be enabled).
4 Message Available	16	Data is available in the instrument's output buffer.
5 Standard Event	32	One or more bits are set in the Standard Event Register (bits must be enabled).
6 Master Summary	64	One or more bits are set in the Status Byte Register (bits must be enabled).
7 Standard Operation	128	One or more bits are set in the Standard Operation Register (bits must be enabled).

Il complesso sistema di bit di stato dello standard IEEE-408.2

Lo standard IEEE-488.2 prevede una miriade di registri di stato, tra cui quelli che indicano che il comando è completato, che non si sono verificati errori, oppure che ci sono errori in

coda da prelevare successivamente. Nessuno di questi registri di stato è stato implementato nella macchina. Lo standard prevede pure una complessa struttura di accodamento degli errori e numerosi flag di segnalazione di stati interni. La presenza di un errore viene segnalata da un flag, ed è cura del programmatore recuperare successivamente il codice e il testo degli errori accodati con comandi successivi di interrogazione. Questa struttura si presta bene ad essere utilizzata proficuamente su bus sofisticati come il GPIB, ma è poco adatta per una connessione punto-a-punto come quella realizzata dalla seriale.

Per quanto riguarda i messaggi di errore si è preferito quindi allontanarsi dallo standard, e prevedere che il controller rispondesse al PC, dopo ogni comando ricevuto, con un codice numerico decimale a due cifre, uno spazio vuoto e una stringa informativa, in modo da rendere molto facile l'interpretazione dell'errore sia ad un operatore umano (che può leggere facilmente la stringa informativa) sia ad una macchina (che può leggere facilmente il codice numerico, in posizione fissa all'inizio della stringa restituita, e di lunghezza costante pari a due caratteri). Del resto anche gli errori dello standard prevedono un codice numerico e una stringa descrittiva. Ciò che cambia è la modalità di comunicazione di questi errori: immediata nel mio circuito, ad accodamento secondo lo standard.

501	I/O processor: isolator framing error
502	I/O processor: isolator overrun error
511	Communications: RS-232 framing error
512	Communications: RS-232 overrun error
513	Communications: RS-232 parity error
514	RS-232 only: unable to execute using HP-IB There are three commands which are allowed only with the RS-232 interface: SYSTem:LOCal, SYSTem:REMote, and SYSTem:RWLock.
521	Communications: input buffer overflow
522	Communications: output buffer overflow
532	Not able to achieve requested resolution The instrument cannot achieve the requested measurement resolution. You may have specified an invalid resolution in the CONFigure or MEASure? command.

Alcuni dei messaggi di errore restituiti dallo strumento Agilent 34970A.
Un certo numero di questi sono sostanzialmente identici a quelli restituiti da Relais Matrix.

Messaggi informativi

01 Completed

Il comando è stato riconosciuto ed eseguito con successo. Alcuni comandi sono senza effetto (per esempio *CLS) oppure senza effetto immediato, (ad esempio un comando che cambia

SubSystem di default soltanto, come SYSTEM o ROUTE). Anche in questi casi si otterrà il messaggio *01 Completed*, come verifica che il comando è stato correttamente riconosciuto.

02 Hallo

Messaggio stampato all'accensione, oppure a seguito del comando *RST. All'accensione, e in seguito ad un comando *RST, viene eseguita una routine di autotest (analoga a quella eseguita in risposta alla query *TST?), durante la quale vengono inviati dei pattern di bit agli shift register presenti sui moduli, al fine di determinare quanti ne siano collegati, o l'eventuale interruzione della catena. Il messaggio *02 Hallo* viene inviato solamente se la routine di autotest ha successo, altrimenti viene inviato uno dei due messaggi di errore *09 Interrupted chain* oppure *15 Too many modules*.

Se si riceve un messaggio *02 Hallo* in circostanze differenti, è segno che il circuito si è spento e riacceso oppure il solo PIC resettato per problemi di alimentazione, e tutte le configurazioni (stato dei relais, Subsystem corrente) sono andate perdute. In questo caso *02 Hallo* rappresenta un messaggio di errore indice di un problema hardware.

Questo messaggio è stato previsto infatti proprio per segnalare la mancanza di alimentazione o problemi di reset spuri del microcontrollore dovuti a malfunzionamenti hardware, anche se può essere utilizzato per rilevare automaticamente l'accensione dell'apparecchio.

Errori del parser

03 Unimplemented

Il commando indicato non è implementato in questa macchina, ma viene comunque processato dal parser in modo da dare all'utente una indicazione che il comando è stato scritto correttamente, anziché mostrare il generico messaggio di errore *04 Unknown Header*. Così procedendo è più facile inserire in revisioni successive del firmware il codice che esegue i comandi previsti ma non ancora implementati, dei quali è stato però codificato un *place holder*. Una delle caratteristiche non implementate segnalate è l'uso del carattere ":" (due punti) nelle liste di canali.

04 Unknown Header

L'identificatore inviato non fa parte del set di istruzioni del controller.

05 Invalid char

E' stato incontrato un carattere non valido (non facente parte della sintassi) oppure uno valido è stato incontrato al posto sbagliato.

Si noti che il parser, in caso di sintassi errata, fornisce sempre e solo uno dei due errori *04 Unknown Header* o *05 Invalid char*. Si potrebbero includere messaggi di errore più mirati, che faciliterebbero il debug di comandi errati inviati al controller, a costo di una occupazione della memoria FLASH del PIC maggiore per memorizzare le differenti stringhe di errore. Non occorre riscrivere il parser, che internamente già distingue tra numerosi tipi di errori, ma è costretto a segnalare all'esterno sempre e solo uno dei due codici a lui riservati disponibili nella routine di segnalazione errori.

Specificazione canale non valido o non presente

06 Slot out of range

E' stato messo un numero di slot pari a 500 600 700 800 o 900. Il controller supporta al massimo 4 slot, numerati 100, 200, 300, 400.

07 Chan out of range

E' stato messo un numero di canale non corretto. Riga e colonna devono essere compresi tra 1 e 4, pertanto i canali disponibili sono 16, numerati 11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34, 41, 42, 43, 44. E' stato seguito lo stesso sistema di numerazione della macchina Agilent di riferimento.

08 No module in slot

Non c'è nessun modulo montato nello slot richiesto. Es.: è stato chiesto di chiudere il relais 312 (prima riga, seconda colonna, sul modulo montato nello slot 3), ma sono stati collegati solo due moduli al controller, ai quali sono stati assegnati i numeri di slot 100 e 200.

Errori hardware nella catena di shift-register

09 Interrupted chain

La catena degli shift-register è fisicamente interrotta o non terminata bene col ponticello di feedback o qualcosa nell'hardware non funziona bene.

15 Too many modules

Sono stati collegati 5 o più moduli in cascata, oppure qualcosa nell’hardware non funziona bene. Si ricorda che il controller accetta al massimo 4 moduli di relais, che vanno collegati in cascata; sull’ultimo occorre inserire il terminatore che collega il dataout dell’ultimo shift-register al piedino feedback del microcontrollore.

La catena viene controllata esclusivamente all’accensione, dopo un *RST, o dopo un autotest *TST?. Questo errore può essere visualizzato solo in uno di questi 3 casi. Se si aggiungono o rimuovono moduli a circuito acceso, i risultati sono imprevedibili: i relais possono anche accendersi e spegnersi apparentemente a caso, poiché il controller crede che sia collegato un certo numero di moduli mentre in realtà ne è collegato un numero diverso, e può aggiornare gli shift-register in maniera errata. E’ in generale fortemente sconsigliato collegare moduli a caldo. Si suggerisce di procedere sempre in questo modo: spegnere il circuito, collegare i moduli, inserire il terminatore sull’ultimo, alimentare il circuito.

Errori nella richiesta di salvataggio/recupero stato di memoria

10 Empty state

La posizione di memoria è vuota e non può essere richiamata con un *RCL

11 Different modules

C’è un numero diverso di moduli negli slot rispetto a quando si è memorizzato lo stato nella posizione di memoria. Il comando *RCL viene rifiutato.

Questi messaggi di errore, pur codificati nella routine di segnalazione errori, non vengono mai visualizzati con la revisione corrente del firmware perché le funzionalità di salvataggio e recupero stati di memoria non sono state ancora implementate.

Errori nella comunicazione seriale

12 RS232 framing

Errore di trama nella comunicazione Rs232. Si attende il successivo new-line prima di continuare a processare l’input. Il led rosso rimane acceso fino all’arrivo del new-line. Se il carattere che ha subito un errore di trama è proprio il new-line, l’intera riga successiva verrà ignorata.

13 RS232 overrun

Errore di overrun nella comunicazione Rs232. La UART del PIC memorizza due caratteri in un buffer FIFO, e il terzo carattere nel registro dell'ultimo carattere arrivato. Se arriva un quarto carattere senza che il firmware utente svuoti il buffer FIFO, si genera l'errore di overrun. Si attende il successivo new-line prima di continuare a processare l'input. Il led rosso rimane acceso mentre il PIC non processa l'input, in attesa di un carattere newline.

Stringhe memorizzate internamente come messaggi di errore

14 PaoloSancono,RelaisMatrix,1.0,1.0

Questo messaggio viene visualizzato, senza il 14 e senza lo spazio in testa, in risposta al comando *IDN? Internamente viene conservato assieme ai messaggi di errore per riutilizzare la routine che stampa stringhe sulla seriale, ottimizzando il codice.

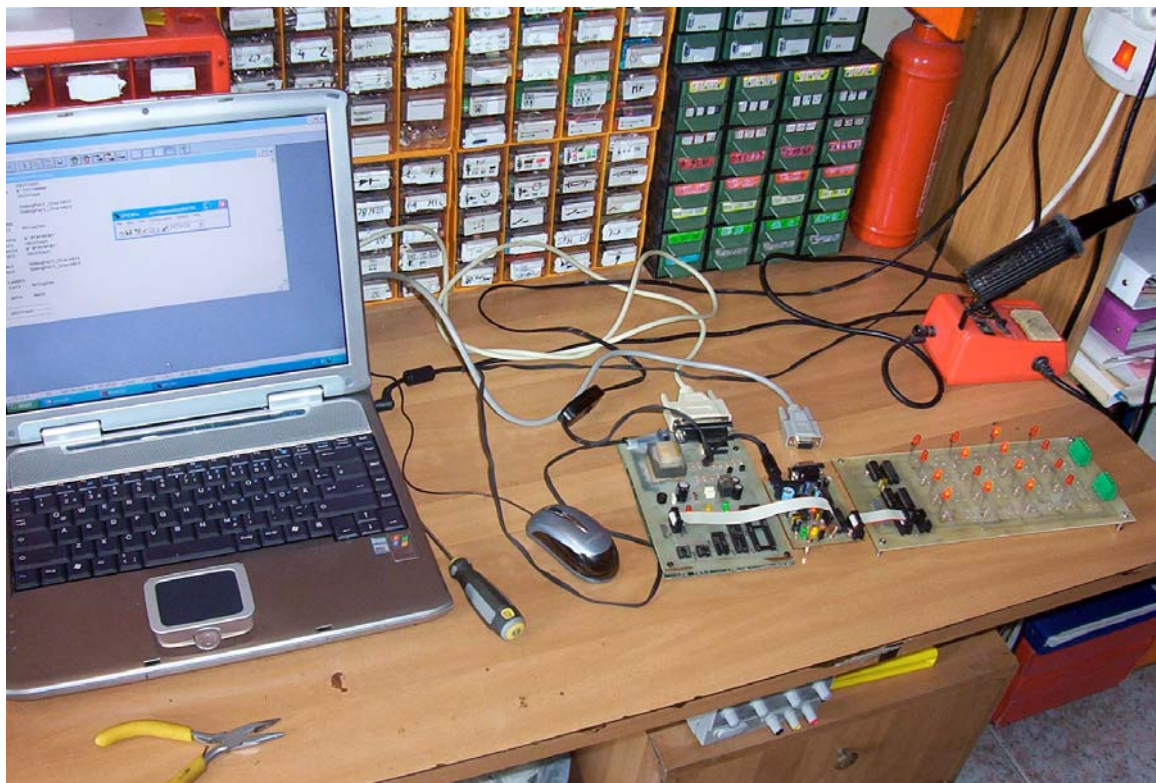
<table>
<tr><td>Messaggi informativi
01 Completed
02 Hallo</td><td>Errori hardware
09 Interrupted chain
15 Too many modules</td></tr>
<tr><td>Errori del parser
03 Unimplemented
04 Unknown Header
05 invalid char

Canale non valido o non presente
06 Slot out of range
07 Chan out of range
08 No module in slot</td><td>Errori memoria (*SAV e *RCL)
10 Empty state
11 Different modules

Errori nella comunicazione seriale
12 RS232 framing
13 RS232 overrun</td></tr>
</table>

Tabella riassuntiva messaggi di errore generabili

Capitolo 4 - Firmware

Stazione di lavoro

La workstation di sviluppo del firmware al completo: PC portatile con MpLab ed EpicWin in esecuzione, programmatore, controller, modulo relais senza i relais montati, tutto cablato.

La stazione di lavoro per la stesura e verifica del firmware è costituita da un PC portatile (più indicato per l'interazione con i circuiti, rispetto ad un PC da tavolo), il programmatore collegato alla porta parallela, la scheda controller, collegata alla porta seriale, con un modulo inserito.

Prima di collegare contemporaneamente la porta parallela e la porta seriale a due circuiti elettrici che condividevano la massa tramite l'ICSP, è stato verificato che anche il PC fornisse lo stesso potenziale di massa sulla porta parallela e sulla porta seriale. Per fortuna era così, altrimenti si sarebbe dovuto procedere all'isolamento galvanico di uno dei dispositivi o del collegamento ICSP, poiché non era proponibile di scollegare e ricollegare continuamente i dispositivi: il lavoro di debug sarebbe risultato estenuante, e alla prima distrazione si sarebbe rotta una porta del PC. Per fortuna è stato invece possibile tenere tutto permanentemente collegato.

Il PC portatile era sprovvisto di una porta seriale, si è allora usato un adattatore USB-Seriale, con tutti i vantaggi che ne derivano: se serve una porta in più, basta comprare un altro

adattatore. Se il software sul PC blocca la porta per un errore di programmazione, non è necessario riavviare il PC per riprenderne il controllo, ma è sufficiente scollegare e ricollegare l'adattatore per liberare le risorse impegnate e riconfigurare la nuova porta, grazie alla tecnologia plug-and-play[51].

Come si nota in figura i relais non erano presenti durante lo sviluppo del firmware. Questo per diversi motivi:

- dovevano ancora essere acquistati a prezzo conveniente ad una fiera locale di elettronica anziché a prezzo pieno al negozio di componenti elettronici;
- si voleva testare il circuito con meno incognite contemporanee per volta, e carichi induttivi possono procurare malfunzionamenti difficili da rilevare, quindi inizialmente si è preferito testare il resto delle funzionalità;
- i relais dopo aver fatto "klic" "klac" tante volte, fanno "kloc" e si rompono: vale a dire che hanno un certo numero di azionamenti nella loro vita utile, e tendono a scaldarsi e rompersi rapidamente se li si fa commutare ripetutamente e rapidamente; nelle prime revisioni del firmware non erano da escludere commutazioni spurie e rapide a causa di errori nel software o nell'hardware (tipico il caso di segnali lasciati flottanti, letti come sequenze casuali di "0" e "1" da circuiti digitali).

All'inizio la basetta del modulo di espansione è stata quindi testata con i soli led montati.

Controllo di flusso tramite macro: psifthen 1.2

Macro e sottoprogrammi

Il compilatore assembler usato consentiva l'uso di macro, vale a dire la possibilità di arricchire la sintassi con ulteriori comandi, ai quali è possibile passare parametri, che vengono espansi nelle righe che li costituiscono, dopo aver sostituito i parametri con i valori immediati. Una macro si differenzia da un sotto-programma poiché se è presente N volte nel codice, ed è costituita da M istruzioni, verrà espansa in NxM istruzioni, mentre un sottoprogramma occuperà la memoria una volta sola. Ai sottoprogrammi assembler è possibile passare però un unico parametro, e con forti limitazioni, attraverso l'accumulatore a 8 bit della CPU: se si desidera passare più parametri, occorre gestire, senza aiuto particolare da parte del compilatore, alcune variabili in RAM da utilizzare per il passaggio dei parametri.

---

[51] Scherzosamente detta plug-and-pray (letteralmente: collega e prega) poiché quando le cose non vanno bene, senza questa tecnologia era possibile tentare di riconfigurare i jumper hardware, con questa tecnologia, in cui i jumper sono stati eliminati, non resta altro che pregare.

Con l'esperienza si capisce quando è opportuno usare una macro e quando un sottoprogramma.

Il controllo di flusso

Una grave pecca dell'assembler, rispetto a linguaggi di alto livello, è la mancanza di strutture avanzate per il controllo del flusso del programma. Vale a dire l'assenza delle strutture if... then... else..., for..., repeat..., while... presenti nella quasi totalità dei linguaggi di programmazione. In assembler strutture di questo tipo si implementano con una schiera di salti incondizionati e condizionati, difficile da seguire per lo sviluppatore e possibile fonte di errore.

Per ovviare a questo inconveniente, ho scritto una serie di macro, raccolte nel file psifthen.inc[52], da includere nel proprio file asm mediante la direttiva al compilatore

```
#INCLUDE <psifthen.inc>
```

Le macro disponibili consentono di verificare diverse condizioni, orientate a byte oppure bit, e vengono espanse in un codice molto compatto (2 o 3 opcode per istruzione).

Uno degli inconvenienti di queste macro è che costringono il programmatore a dichiarare un identificatore di etichetta (utilizzata per i salti condizionati) unico per ogni struttura di cui si ha bisogno, non essendoci la possibilità, per le macro, di richiedere al compilatore di creare una etichetta casuale senza chiedere all'utente di specificarne una esplicitamente. Questo complica un po' le operazioni di copia/incolla di pezzi di codice, poiché occorre modificare gli identificatori duplicati. L'inconveniente è ben tollerato data la ricchezza espressiva possibile che rende molto più comprensibile il codice.

Riporto qualche esempio di utilizzo, e una succinta descrizione delle macro presenti, rimandando i dettagli alla documentazione interna, nel file psifthen.inc, il cui listato integrale è presente in appendice.

Struttura if-then-else

Esempio di utilizzo if-then con condizione di non uguaglianza (not equal, ne)

```
ifne   Data1,Data2,myif
       ;<istruzioni>
       ;<istruzioni>
ifend  myif
;<resto del programma>
```

[52] La raccolta di macro, da me realizzata, viene distribuita con licenza GPL. Il file .inc è giunto, al momento in cui si scrive, alla revisione 1.2, ed è liberamente scaricabile dal mio sito http://www.ideegeniali.it/. Il file può essere ridistribuito gratuitamente e utilizzato nei propri progetti a patto di distribuire anche i lavori derivati con licenza GPL.

Dove myif è l'etichetta utilizzata per le condizioni, Data1 e Data2 due file registers da confrontare. Sono disponibili altre macro relative agli if che realizzano i controlli di minoranza, di maggioranza, potendo confrontare tra loro valori immediati, file register e l'accumulatore nelle varie combinazioni possibili.

E' disponibile anche il costrutto if-then-else, ma occorre, in tal caso, indicare due etichette, così:

```
iflt          Data1,Data2,if1
              ;istruzioni per condizione vera
ifelse if1,if2
              ;istruzioni per condizione falsa
ifend         if2
```

Un costrutto del tipo AND tra due condizioni si realizza semplicemente inserendo due costrutti if uno sotto l'altro, ed è possibile riutilizzare la stessa etichetta, così:

```
ifgtel data,min,myif
ifltl         data,max+1,myif
              ;<istruzioni>
              ;<istruzioni>
ifend         myif
```

Dove il file register data viene confrontato con i due valori immediati min e max per verificare che sia compreso nell'intervallo. Le macro implementano gli if mediante il salto condizionato all'etichetta rappresentata da ifend. Il salto viene effettuato quando la condizione è falsa; se la condizione è vera, il salto non viene effettuato e le istruzioni comprese tra l'if e ifend vengono eseguite. E' quindi possibile scrivere un salto condizionato utilizzando le macro if con la condizione negata rispetto a quella originaria. Un codice di salto condizionato all'uguaglianza, anziché essere espresso in questo modo:

```
ifee          pippo,paperino,if001
              goto   salta
ifend         if001
```

può essere espresso in forma più compatta in quest'altro modo, con la condizione opposta:

```
ifne          pippo,paperino,salta
```

Per le etichette si è utilizzato sempre lo stesso carattere (la lettera i) seguito da tre numeri, in modo da non dover inventare etichette differenti ogni volta, e indicare chiaramente che le etichette del tipo ixxx hanno l'unico scopo del controllo di flusso tramite macro, e non qualche significato specifico particolare (come per le etichette utilizzate per chiamare sottoroutine).

I mnemonici a due o tre caratteri che seguono i caratteri if, nei nomi delle macro implementate, rappresentano:

- ifee (EE = Equal) ugualianza

- ifne (NE = Not Equal) disegualianza
- ifgte (GTE = Greater Then or Equal) maggioranza o equalianza
- iflt (LT = Less Than) minoranza stretta

I confronti sono normalmente tra due file register. Volendo realizzare confronti tra un file register e il valore dell'accumulatore o un valore immediato, si fa seguire agli mnemonici precedenti la lettera W per l'accumulatore o l (literal) per un valore immediato.

- ifeew (EEW = Equal W)
- ifnew (NEW = Not Equal W)
- ifgtew (GTEW = Greater Than or Equal W)
- ifltw (LTW = Less Than W)
- ifeel (EEL = Equal Literal)
- ifnel (NEL = Not Equal Literal)
- ifgtl (GTL = Greater Than Literal)
- ifltl (LTL = Less Than Literal)

Sono state poi scritte macro per la verifica di appartenenza ad un intervallo espresso mediante due valori immediati:

- ifr (R = Range) File register nel range specificato
- ifrw (RW = Range W) Accumulatore W nel range specificato

Altre macro implementano condizioni orientate ai bit anziché ai byte:

- ifbs (BS = Bit Set) Bit a 1
- ifbc (BC = Bit Clear) Bit a 0

Le macro sono implementate sostanzialmente con istruzioni di sottrazione e verifica del segno del risultato, sfruttando la ALU del PIC. Ecco ad esempio la macro per il controllo dell'ugualianza:

```
ifee   macro  file1,file2,alabel  ;if egual (File-File)
       movf   file1,W
       subwf  file2,W
       btfss  STATUS,Z
       goto   alabel
endm
```

Viene caricato in W il valore del primo file register, sottratto quindi il secondo, lasciando il risultato in W per non sovrascrivere i file registers, ed effettuato il salto condizionato se il risultato non è zero. Il flag Z del registro di stato dell'ALU del PIC è mappato in RAM e la CPU accede a questo e ad altri registri di controllo come accederebbe ad un file register, senza

la necessità di istruzioni apposite per la gestione di tali flag.[53] L'opcode *btfss*, che può disorientare chi non conosce l'assembler dei PIC, è un acronimo, e sta per "Bit Test File, Skip Set", ovvero "Controlla un bit nel file register, salta l'istruzione seguente se il bit è a 1". Assieme all'altra istruzione sorella, *btfsc* (Bit Test File, Skip Clear) rappresenta l'unico modo per effettuare salti condizionati nell'assembler dei PIC. Sono istruzioni potenti, in quanto consentono di controllare se un bit è settato o no senza ricorrere alle maschere con AND, come occorre fare in linguaggi assembler di altre CPU.

Altre macro utili

Nel file psifthen.inc trovano posto anche altre macro utili.

- dnop Attende il tempo di due nop, ma occupa una sola word in memoria
- qnop Attende il tempo di quattro nop, ma occupa una sola word in memoria
- let Assegna il contenuto di un file register ad un altro
- letl Assegna un valore immediato ad un file register
- for Realizza cicli for a decrescere, a partire dal valore di un file register
- forl Variante per iniziare il conteggio da un valore immediato
- forw Variante per iniziare il conteggio dal valore attuale dell'accumulatore
- next Conclude il ciclo for
- repeat Assieme alle macro if, realizza strutture repeat-until
- until Chiude la struttura repeat-until
- wile Realizza la struttura while-do (While è parola riservata di MPASM)
- wilend Chiude la struttura while-do
- negbit Inverte lo stato di un bit indicato esplicitamente con un valore immediato
- eeread Legge un byte dalla EEPROM dati del PIC
- eewrt Scrive un byte dalla EEPROM dati del PIC
- eewait Attende che un ciclo di scrittura su EEPROM si concluda

La differenza tra la struttura repeat-until e while-do è, come in Pascal, che nel primo caso si itera almeno una volta anche in caso di condizione falsa fin dall'inizio. Le macro sono state citate tutte perché vengono utilizzate nel firmware e, non conoscendole, è impossibile seguire il codice. Non è possibile parlare in maniera più approfondita dell'assembler e di tali macro in questa relazione per mancanza di spazio.

---

[53] A titolo di paragone, in altre CPU il registro di stato è interno ed esistono istruzioni specifiche per il salto condizionato al valore del flag di Zero e del flag di Carry, che indicano

Bios

Differenze tra firmware e software

Realizzare il firmware per un microcontrollore può essere profondamente differente e ben più difficile rispetto a realizzare un programma al PC. Finché il firmware non è pronto, non si può verificare la funzionalità dell'hardware; viceversa, se l'hardware non funziona, anche un software scritto bene non produrrà gli effetti sperati. Se il circuito ha poi pochi o nessun dispositivo di segnalazione, è veramente difficile valutare il buon funzionamento del programma, poiché si è costretti ad analizzare i segnali elettrici ai morsetti esterni dell'integrato.

"Debug hardware" è una buona dizione per indicare i primi passi con il firmware e l'hardware, quando non funzionano entrambi contemporaneamente, ed è molto difficile trovare i problemi, poiché le variabili in gioco sono tante: se un led non si accende, può darsi che sia bruciato il led, che ci sia una pista interrotta, che sia un collegamento ad una pista errata, che la sorgente di clock per il microcontrollore non stia oscillando, che il microcontrollore abbia tensione di alimentazione insufficiente, che il piedino di master reset sia attivo, oppure che le routine di inizializzione non abbiano configurato il piedino corretto come uscita, che il flusso del programma non sia dove ci aspettiamo, che disturbi in radiofrequenza sovrascrivano i registri interni del PIC, e moltissime altre questioni, da valutare con pazienza, che non si verificano quando si sviluppa un software per PC, in cui almeno tastiera, mouse e monitor funzionano fin dall'inizio.

Un buon punto di inizio

Le prime routine da programmare sono quindi quelle relative ai sistemi di Input/Output più semplici presenti sulla basetta. Nel nostro caso, il led rosso di segnalazione. Una volta scritta una routine funzionante per il Led, si potrà usare questa segnalazione per sviluppare il resto, segnalando con il led il passaggio per punti critici del codice, una delle caratteristiche più necessarie durante il debug.

Man mano, quindi, si realizza e si testa il BIOS[54] del firmware. Si è quindi obbligati ad una programmazione bottom-up, ovvero piccole routine funzionanti di accesso all'hardware, che

---

se l'ultima operazione ha avuto per risultato Zero o ha generato un riporto.

[54] Basic Input Output System. Letteralmente: Sistema base di ingresso/uscita, ovvero di comunicazione del firmware con l'hardware. In un PC, il BIOS è già pronto in una memoria non volatile sulla scheda madre, e difficilmente un utente riprogramma da sé il BIOS. In un circuito con microcontrollore, il BIOS va realizzato ex-novo ogni volta.

costruiranno poi le routine più complesse. E’ essenziale avere routine affidabili per le periferiche di output poiché consentiranno di comunicare con il mondo esterno e di testare il resto del codice.

Inizializzazione

Prima ancora di scrivere una riga di codice, occorre configurare il microcontrollore per l’utilizzo della sorgente di clock corretta e per l’attivazione o disattivazione di alcune funzionalità avanzate programmando correttamente la configuration word[55]: se non si configura la configuration word correttamente per l’utilizzo di un cristallo come sorgente di clock, nel nostro circuito non partiranno neppure le oscillazioni del quarzo, poiché il relativo circuito di eccitazione sarà spento. In queste condizioni è impossibile eseguire istruzioni.

```
        LIST      p=16f628
        #INCLUDE <p16f628.inc>
        #INCLUDE <psifthen.inc>
        __CONFIG _BODEN_ON & _CP_OFF & _DATA_CP_OFF & _PWRTE_OFF & _LVP_OFF & _MCLRE_OFF &
_HS_OSC
```

All’accensione tutti i piedini del PIC sono in stato di alta impedenza. Occorre configurare correttamente i piedini da utilizzare come uscite, e gli altri “gadget hardware” presenti all’interno del PIC, selezionando le funzioni desiderate per i piedini che hanno un multiplexer tra funzioni differenti. Questo viene svolto dalla routine init nel nostro progetto, la quale a sua volta fa uso di costanti dichiarate all’inizio del programma che riflettono come è stato cablato il circuito e cosa è collegato a ciascun piedino.

```
init    movlw   0x20            ;Cancella i file register da 0x20 a 0x7f
        movwf   FSR             ;in modo da assicurare che tutte le variabili
initL   clrf    INDF            ;del programma siano a zero inizialmente
        incf    FSR,F           ;
        btfss   FSR,7           ;
        goto    initL

        movlw   TrisPortA       ;Imposta i TRIS register per le due porte
        tris    PORTA           ;
        movlw   TrisPortB       ;
        tris    PORTB           ;
        movlw   B'00000111'     ;Configura PORTA come Digital I/O
        movwf   CMCON           ;xxxxx111 Comparator disable, RBA0-3 as Digital I/O
        clrf    PORTA           ;Clear port latches
        clrf    PORTB           ;

        bsf     STATUS,RP0      ;Bank 1
        movlw   B'00100110'     ;Configura USART
        movwf   TXSTA           ;Trasmit Status and Control: 8-bit receive,
                                ;Trasmit enable, Asincronous mode, Baud Rate Generator HS
        movlw   .20             ;Configura USART
        movwf   SPBRG           ;Baud Rate Generator Register  10: 113636 (115200)
                                ;20: 59524 (57600) @ 20 MHz FOsc
        bcf     STATUS,RP0      ;Bank 0
        movlw   B'10010000'     ;Configura USART
        movwf   RCSTA           ;Receive Status and Control: Serial Port Enable,
                                ;8-bit receive, Continuous receive enable
```

[55] Un registro della memoria FLASH del PIC che configura alcune funzionalità interne.

Il codice cancella i file register utilizzati dal programma, assicurando il valore iniziale 0, imposta la direzione di ogni piedino, disattiva i comparatori analogici, rendendone disponibili piedini in multiplex con PORTA, porta digitale di I/O d'uso generico ad 8 bit. Viene infine configurata la USART del PIC con le velocità desiderate e senza l'uso di interrupt. Consultare il data-sheet del PIC16F628 per informazioni sul significato di ciascun bit settato nei registri di controllo.

Led, Buzzer, linea CTS della seriale

Per l'accensione e lo spegnimento del Led, del Buzzer e della linea CTS della porta seriale, sono state scritte alcune macro da una sola istruzione: è infatti sufficiente cambiare lo stato di un bit della porta corretta per utilizzare questo semplice hardware, a patto che tutto sia stato configurato a dovere.

```
TrisPortA       equ     B'11110010'
TrisPortB       equ     B'11000011'

ShRegPort       equ     PORTA
ClockBit equ    0
FeedbackBit     equ     1
StoreBit equ    2
DataBit         equ     3

CtsPort         equ     PORTB
CtsBit          equ     0
RtsPort         equ     PORTB
RtsBit          equ     3

BuzzPort equ    PORTB
BuzzBit         equ     4

LedPort         equ     PORTB
LedBit          equ     5

LedOn   macro
        bcf     LedPort,LedBit
        endm

LedOff  macro
        bsf     LedPort,LedBit
        endm

LedToggle macro
        negbit  LedPort,LedBit
        endm

BuzzOn  macro
        bcf     BuzzPort,BuzzBit
        endm

BuzzOff macro
        bsf     BuzzPort,BuzzBit
        endm

RtsGo   macro
        bsf     RtsPort,RtsBit
        endm

RtsHalt macro
        bcf     RtsPort,RtsBit
        endm
```

Sono state dichiarate costanti che rispecchiano i collegamenti hardware, e sono stati scelti nomi evocativi per le macro, in modo che il codice che le usa sia molto chiaro.

Volendo funzioni di segnalazione più avanzate, per esempio led lampeggiante, oltre che fisso acceso e fisso spento, conviene isolarne il codice in apposite routine dal nome appropriato e utilizzare i “gadget hardware” disponibili sul PIC, ad esempio i contatori in grado di generare interrupt quando sono in stato di *overflow*, rendendo del tutto indipendente dal flusso del programma principale l’accensione e lo spegnimento del Led.

Seguendo il consiglio di isolare il codice per ciascuna funzionalità, anche semplice, di cui si ha bisogno, è stato isolato il codice per realizzare un beep col buzzer nella routine dal nome evocativo BuzzBeep: anche se questa routine viene chiamata da un unico altro punto del codice (la routine di gestione errori), è comunque più leggibile il comando call BuzzBeep che non l’inserimento delle quattro istruzioni necessarie a realizzare il beep in sequenza, all’interno della routine di gestione errore.

```
; --------------------
; ----- BuzzBeep -----
; --------------------
; Effettua un beep accendendo il Buzzer, attendendo 10 centesimi di secondo, e spegnendolo

BuzzBeep
        BuzzOn
        movlw   .10
        call    DelayCs
        BuzzOff
        return
```

La routine a sua volta utilizza DelayCs, che non descriverò, rimandando alla documentazione interna del codice, che realizza una pausa di W (il valore passato con l’accumulatore) centesimi di secondo. Utilizzando nomi opportuni per le routine è possibile scrivere codice leggibile anche in assembler. BuzzBeep è stata scritta come routine, anziché come macro, poiché era più lunga di una istruzione, e scriverla come macro avrebbe aumentato, in generale, l’occupazione in memoria del programma. Nel nostro caso, siccome viene richiamata un’unica volta, in realtà così procedendo si è sprecata una word di memoria per l’istruzione return, ma si è guadagnato in leggibilità del codice.

Shift Register

Appena più difficile è stato gestire gli shift-register, e i relais lì connessi. Una prima routine shiftout, che usa due variabili temporanee ShiftMe e shiftoutLooper, invia gli 8 bit presenti nell’accumulatore W sul bus di sistema, dal meno significativo al più significativo.

Per le variabili temporanee delle routine, si è deciso di sprecare un po’ di memoria dichiarandone una per ciascuna routine, anziché riutilizzare sempre le stesse posizioni in memoria, col rischio di pericolose sovrapposizioni e la necessità di una analisi approfondita

dello *scope*[56] delle variabili, del quale, in assembler, si deve fare carico il programmatore e non il compilatore.

```
;-------------------------
;----- shiftout store -----
;-------------------------
; Coppia di routine di basso livello per interazione con gli shift-registers
; shiftout
;   Manda gli 8 bit presenti in W agli shift register, dal LST al MSB, con trasmissione
sincrona,
;  tramite i pin data e clock
; store
;  Manda un fronte di salita sugli store degli shift register, trasferendo l'informazione
;  dagli shift register ai latch, agli stadi di uscita, ai darlington, ai relais e led

shiftout
        movwf   ShiftMe
        forl    shiftoutLooper,.8,i012
                bcf     ShRegPort,DataBit
                btfsc   ShiftMe,0
                bsf     ShRegPort,DataBit
                bcf     ShRegPort,ClockBit
                bsf     ShRegPort,ClockBit
                rrf     ShiftMe,F
        next    shiftoutLooper,i012
        bcf     ShRegPort,DataBit       ;E' bene che stia sempre a zero
        return
```

La routine fa uso delle macro per le strutture for-next descritte in precedenza, e dell’istruzione rrf (Rotate Right File) che sposta verso destra i bit presenti in un file register, rendendo disponibile nel bit meno significativo, via, via, gli 8 bit iniziali.

```
store
        bcf     ShRegPort,StoreBit
        bsf     ShRegPort,StoreBit
        return
```

La routine store invia un fronte di salita al segnale di store, trasferendo lo stato degli shift register ai latch annessi. Finché non si richiama store, gli shift-register vengono aggiornati, ma i latch conservano la configurazione precedente per i relais.

Lo configurazione dei Relais per 4 moduli viene memorizzata compattamente in 8 byte di Ram, nella variabile Relais. Per accedere ad una variabile larga più byte, si fa ricorso all’unica modalità di indirizzamento indiretto del PIC, ovvero all’uso dei registri mappati in ram FSR e INDF (File Select Register e Indirect File): nel primo si seleziona la locazione a cui accedere in modo indiretto, nel secondo si scrive o legge il valore della locazione. La routine aggiornaModuli trasferisce il contenuto della variabile Relais agli shift register, sfruttando shiftout e store descritte precedentemente.

```
;-------------------------
;----- aggiornaModuli -----
;-------------------------
; Aggiorna i relais sulle schede con i contenuti attuali della variabile (a 8 byte) Relais,
; inviando i bit di Relais, nell'ordine opportuno, agli shift-register montati sui moduli

aggiornaModuli
        movlw   Relais
        addlw   .7
        movwf   FSR
        forl    aggiornaModuliLooper,.8,i034
```

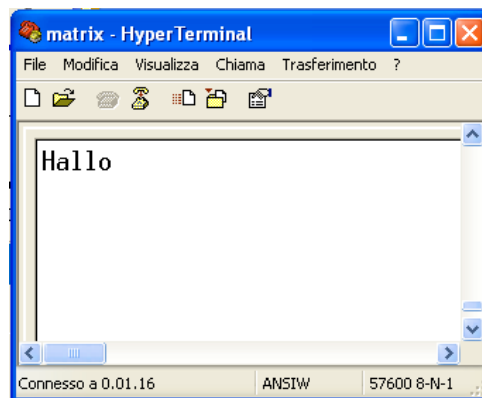
[56] La durata nell’ambito del programma.

```
                movf     INDF,W
                call     shiftout
                decf     FSR,F
        next    aggiornaModuliLooper,i034
        call    store
        return
```

Vengono inviati per primi i bit relativi ai moduli più lontani nella catena, poi man mano quelli per i più vicini, per la maniera nella quale funzionano gli shift register.

Il programma aggiorna ogni volta 4 moduli e 256 relais, confidando nel fatto che il numero di moduli presenti, di cui ha fatto il test all'inizio, sia quello giusto, e non verranno interessate dall'operazione schede assenti, sulle quali viene buttata "spazzatura". E' necessario, pertanto, che i moduli non vanno inseriti o rimossi "a caldo", e comunque non vanno utilizzati se non prima di una riaccensione, un *RST, o un *TST?, per evitare condizioni di funzionamento imprevedibili.

Porta seriale

La prima prova effettuata con la USART del PIC è stata implementare il classico "Hallo world"[57], ovvero inviare la stringa "Hallo" sulla porta seriale leggendola tramite il Terminale di Windows.

Il programma "Hallo world" per l'uso della seriale con il PIC

Grazie alla presenza di una porta UART tra i "gadget hardware" del PIC, le istruzioni per inviare o ricevere un carattere sono banali; un po' meno semplice è la gestione degli errori di trasmissione o ricezione, che descriverò più avanti assieme al ciclo principale del programma. Qui descriverò invece la routine in grado di trasmettere stringhe memorizzate in look-up tables.

```
;---------------------------------------------
;----- trasmit newline sendString sendMsg ----
```

[57] Tradizionalmente, il primo programma realizzato in ogni linguaggio di programmazione è quello che stampa su schermo la scritta "Hallo world!", in cui il programma, appena creato, saluta il mondo intero rendendo manifesta la sua esistenza.

```
;--------------------------------------------
; Routine di trasmissione caratteri sulla seriale
;
; trasmit
;  Passare in W un byte da trasmettere: verrà trasmesso sulla RS232
;  eventualmente subito dopo aver atteso che terminasse la trasmissione precedente,
;  se ancora in corso. In questo caso la procedura è bloccante. Non è stato previsto infatti
;  un buffer di uscita e una gestione ad interrupt, poiché non necessario
;  per il nostro progetto.
;
; newline
;  Invia semplicemente il carattere 0x0A, oppure la coppia 0x0D 0x0A se Flag[FlCrLf] è settato
;
; sendString
;  Invia solo una stringa sulla seriale, senza neppure il carattere new-line dietro.
;  Passare in W il codice BCD del messaggio desiderato
;  (messaggi definiti nelle tavole in pagina 0)
;
; sendMsg
;  Invia un codice numerico a due cifre, uno spazio, un messaggio testuale e un carattere
;   new-line sulla seriale. Passare in W, in BCD, l'indice del messaggio da trasmettere.
;   Se l'indice del messaggio è >=3, verrà emesso un beep dal buzzer. Infatti i codici
;   01 e 02 sono informativi, i codici superiori messaggi di errore

trasmit
        btfss   PIR1,TXIF
        goto    trasmit
        movwf   TXREG
        return

newline
        ifbs    Flag,FlCrLf,i103
                movlw   0x0D
                call    trasmit
        ifend   i103
        movlw   0x0A
        call    trasmit
        return

sendMsg
        movwf   sendMsgBcd
        swapf   sendMsgBcd,W
        andlw   0x0F
        addlw   "0"
        call    trasmit
        movf    sendMsgBcd,W
        andlw   0x0F
        addlw   "0"
        call    trasmit
        movlw   " "
        call    trasmit
        movf    sendMsgBcd,W
        call    sendString
        call    newline
        ifgtel  sendMsgBcd,0x03,i043
                call    BuzzBeep        ;Beep se sendMsgBcd >= 3
        ifend   i043
        return

sendString
        movwf   sendMsgBcd
        clrf    sendMsgTemp
        for     sendMsgLooper,sendMsgBcd,i040
                movf    sendMsgLooper,W
                call    tavMsgSizes
                addwf   sendMsgTemp,F
        next    sendMsgLooper,i040
        incf    sendMsgBcd,W
        call    tavMsgSizes
        forw    sendMsgLooper,i041
                movf    sendMsgTemp,W
                call    tavMsgText
                call    trasmit
                incf    sendMsgTemp,F
        next    sendMsgLooper,i041
        return
```

Trasmit attende che la UART finisca una eventuale trasmissione in corso, e trasmette quindi il carattere passato in W. Se la UART era libera, la routine non è bloccante, se la UART era impegnata, la routine è bloccante. Newline trasmette normalmente il carattere 0x0A. Se è attiva la modalità di debug e test, della quale parlo diffusamente in seguito, fa precedere 0x0A da 0x0D (Carriage return).

SendString trasmette una delle stringhe memorizzate nelle look-up-tables, delle quali parlo più avanti. SendMsg trasmette il codice numerico della stringa, uno spazio, la stringa, e un new-line. Emette inoltre un beep se il messaggio ha indice >=3, ovvero se è un messaggio di errore. Per i messaggi 01 e 02 non viene emesso il beep, poiché sono messaggi informativi.

Data l'architettura Harvard dei PIC, le look-up tables non possono essere memorizzate in RAM. Vengono memorizzate invece in ROM. La CPU accede alla ROM solo per il Fetch delle istruzioni, ma da programma non è dato di accedere ai contenuti della ROM, quindi sembrerebbe impossibile memorizzare look-up tables. Nei PIC esiste però una istruzione apposita: retlw (Return with literal in W) che esegue il ritorno da subroutine lasciando nell'accumulatore un valore immediato. E' l'istruzione ideale per realizzare una look-up table: basta far precedere una lunga sequenza di retlw da una istruzione di incremento di PCL (Program Counter Low, il registro che memorizza gli 8 bit bassi del Program Counter, che conserva l'indirizzo della prossima istruzione da eseguire). Il compilatore facilita ulteriormente il compito sostituendo automaticamente il mnemonico dt (Define Table), che accetta anche una stringa tra doppi apici, con una sequenza di istruzioni retlw.

```
; ----------------------------------
; ----- Tavole (Look up tables) -----
; ----------------------------------

tavMsgSizes
        bcf     PCLATH,0
        bcf     PCLATH,1
        addwf   PCL,F
        dt      .0,.0   ; Uno zero in più perché è comodo nella routine che usa la tavola
                        ; Un altro zero perché il codice 00 non viene utilizzato
        dt      .9,.5,.13,.14,.12,.17,.17,.17,.17
        dt      .0,.0,.0,.0,.0,.0
                        ; Le posizioni 0x0A..0x0F non vengono utilizzate: i messaggi sono
                        ; memorizzati internamente con indici esadecimali, ma non
                        ; utilizzandoli tutti l'approccio è equivalentemente BCD.
                        ; All'operatore esterno gli indici dei messaggivengono descritti
                        ; come decimali. Con questo approccio non sono necessarie
                        ; conversioni decimale-binario
        dt      .11,.17,.13,.13,.33,.16

        IF ((HIGH ($)) != (HIGH (tavMsgSizes)))
                ERROR "La tavola di salto tavMsgSizes ha superato i confini di pagina"
        ENDIF
        IF ((HIGH ($)) != B'00')
                ERROR "La tavola di salto tavMsgSizes ha bisogno di ridefinire PCLATH"
        ENDIF

tavMsgText
        bcf     PCLATH,0
        bcf     PCLATH,1
        addwf   PCL,F
        dt      "Completed"             ;01
        dt      "Hallo"                 ;02
```

```
dt      "Unimplemented"          ;03
dt      "Unknown Header" ;04
dt      "Invalid Char"           ;05
dt      "Slot out of range"      ;06
dt      "Chan out of range"      ;07
dt      "No Module in Slot"      ;08
dt      "Interrupted chain"      ;09
                                 ;0A..0F
dt      "Empty state"            ;10
dt      "Different modules"      ;11
dt      "RS232 Framing"          ;12
dt      "RS232 Overrun"          ;13
dt      "PaoloSancono,RelaisMatrix,1.0,1.0"       ;14
dt      "Too many modules"      ;15

IF ((HIGH ($)) != (HIGH (tavMsgText)))
        ERROR "La tavola di salto tavMsgText ha superato i confini di pagina"
ENDIF
IF ((HIGH ($)) != B'00')
        ERROR "La tavola di salto tavMsgText ha bisogno di ridefinire PCLATH"
ENDIF
```

Notare in coda alla tavola il controllo di pagina di memoria, poiché PCL conserva solo gli 8 bit bassi del program counter, mentre i bit alti sono conservati in PCLATH. Sono state memorizzate separatamente le lunghezze delle stringhe per risparmiare memoria: l’alternativa era il *padding* con spazi alla stringa più lunga. Se occorre ancora più memoria, si può ottimizzare ulteriormente il codice utilizzando un separatore di stringa successiva nel bit più alto, con la condizione di utilizzare solo codici ASCII a 7 bit nei messaggi, ma occorre complicare la routine sendString.

I codici di errore vengono descritti all’utente finale come decimali. Non è stata necessaria però nessuna routine di conversione da decimale a binario o viceversa, poiché internamente i codici vengono memorizzati come esadecimali o BCD, e le posizioni 0A..0F non vengono utilizzate. Questo spreca un po’ di memoria nella tavola, poiché occorre comunque memorizzare degli zeri per la lunghezza delle stringhe non utilizzate, ma risparmia memoria poiché evita routine di conversione decimale binario. Per la quantità e la lunghezza delle stringhe presenti, si è scelto l’approccio ottimale: 6 parole di memoria sprecate per saltare le posizioni 0A..0F, una ventina di parole di memoria risparmiate per l’assenza di una routine di conversione. Se aumenta il numero di stringhe, l’approccio ottimale e quello errato si scambiano di posto.

Stringa Ident

In assembler non è prevista alcuna funzionalità di gestione di stringhe, ovvero sequenze di caratteri di lunghezza variabile dinamicamente durante l’esecuzione del programma. Al nostro programma occorreva questa caratteristica per memorizzare i caratteri ricevuti in sequenza riconoscendo eventuali identificatori presenti. Assieme al BIOS del nostro sistema descrivo quindi anche una rudimentale gestione di stringhe realizzata con due macro e una routine.

```
; -------------------------------
; ----- Variabili e Costanti -----
; -------------------------------
        cblock  0x20
[...]

;--- Variabili globali
Ident:8         ; Buffer che contiene la stringa dell'identificatore in parsing
IdentLength     ; Lunghezza della stringa memorizzata in Ident

[...]

ifident3 macro  char1,char2,char3,alabel ; Si usa come gli altri if in psifthen.inc
        ifeel   IdentLength,.3,alabel    ; Riconosce l'ugualianza tra i primi 3 caratteri di
Ident
        ifeel   Ident,char1,alabel       ; e controlla anche che Ident abbia lunghezza 3
        ifeel   Ident+1,char2,alabel
        ifeel   Ident+2,char3,alabel
        endm

ifident4 macro  char1,char2,char3,char4,alabel   ; Come ifident3 ma controlla 4 caratteri
        ifeel   IdentLength,.4,alabel
        ifeel   Ident,char1,alabel
        ifeel   Ident+1,char2,alabel
        ifeel   Ident+2,char3,alabel
        ifeel   Ident+3,char4,alabel
        endm

[...]

;------------------------
;----- IdentAddChar -----
;------------------------
; Ident è un buffer di byte consecutivi in memoria che implementa in Assembler
; una stringa a lunghezza variabile la cui lunghezza attuale viene conservata in IdentLength
; La routine IdentAddChar aggiunge il carattere passato in W in coda a Ident
; Se Ident è già alla dimensione massima, il carattere successivo viene semplicemente ignorato

IdentAddChar
        movwf   IdentTemp
        incf    IdentLength,F
        ifgtel  IdentLength,.9,i006
                letl    IdentLength,.8
                return
        ifend   i006
        movf    IdentLength,W
        addlw   Ident-1
        movwf   FSR
        let     INDF,IdentTemp
        return
```

Il codice di gestione stringhe prevede un buffer dichiarato come variabile globale che memorizza la stringa, un’altra variabile che conserva la lunghezza attuale dinamica, una routine che aggiunge un carattere in coda alla stringa, l’uso del comando clrf IdentLength per svuotare la stringa settandone la lunghezza a zero e due macro che confrontano i primi 3 o i primi 4 caratteri presenti nella stringa con tre caratteri passati come valore immediato. Queste macro verranno utilizzate dal parser per riconoscere identificatori validi. Il codice scritto con le macro non è efficiente per quanto riguarda l’occupazione in memoria, ma è molto comodo da usare e rende il programma più leggibile. Se la memoria a disposizione dovesse terminare, si può riscrivere la macro in maniera più efficiente come routine, risparmiando in un colpo solo una grande quantità di memoria con un’unica operazione di ottimizzazione (l’ottimizzazione è una delle operazioni più laboriose quando si programma in assembler).

Flag da un bit

Le variabili booleane (un solo bit) vengono memorizzate compattamente in un’unica varibile di un Byte, dichiarata all’inizio del programma in questo modo:

```
Flag            ; Variabili da 1 bit packed in un byte (sotto i singoli bit)

[...]

;--- bits packed in byte Flag
FlVerboseAutotest equ 0  ;Parametro di ingresso per routine autotest:
                         ;settato se deve inviare 1/0 sulla seriale
FlErrorPresent   equ 1   ;Settato se c'è un errore da inviare dopo la ricezione della riga
corrente
FlLogic          equ 2   ;Usato per i test logici composti (and or tra più condizioni)
                         ;nel prog principale
FlEcho           equ 3   ;Settato se è richiesto l'echo di ogni carattere ricevuto sulla
seriale
FlCrLf           equ 4   ;Settato se è richiesto CR/LF anziché LF in uscita
FlExclusive      equ 5   ;Settato se è riconosciuto il tag EXCLusive dopo ROUTe:OPEN o
ROUTe:CLOSe
                         ;siccome è l'unico tag di 3° livello implementato, si è preferito
                         ;un Flag di un bit anziché una apposita variabile di SubSystem
                         ;di 3° livello riconosciuto da un byte
FlInvert         equ 6   ;Parametro di ingresso routine sendINDF
                         ;Settato complementa relais prima di inviare gli 1 e 0 sulla seriale
```

Messaggi di errore

I messaggi di errore sono conservati in due look-up tables, come descritto in precedenza. Per memorizzare le stringhe in maniera efficiente, si sono utilizzati con continuità i codici numerici corrispondenti ai primi numeri interi. Di solito, invece, la strumentazione divide i codici di errore per gruppi, assegnando codici vicini ad errori dello stesso gruppo.

Mentre invia un messaggio di errore, il PIC non presta attenzione ai caratteri ricevuti, e potrebbe verificarsi un rs232 overrun, che verrà segnalato subito dopo aver inviato la propria stringa. Dopo un overrun, il controller attende il successivo new-line prima di continuare a processare l’input.

Anche in caso di altri errori, per esempio *05 Invalid Char*, il controller attende il successivo carattere new-line prima di processare l’input. La condizione di attesa del new-line viene indicata dall’accensione fissa del led, che si spegne non appena si riceve il new-line.

Per i messaggi informativi sono stati utilizzati i primi codici numerici (*01* e *02*) perché, non dovendo produrre il “beep”, era più facile implementare un codice efficiente tenendoli raggruppati all’inizio. E’ stata comunque prevista la chiamata a due routine: una che mostra il messaggio ed effettua il beep, e l’altra che invia semplicemente la stringa, senza neppure precederla dal codice numerico. Questa possibilità è stata usata per inviare la stringa di risposta a *IDN? che, seguendo lo standard, non deve essere preceduta da un codice numerico a due cifre.

```
ifbc    PIR1,RCIF,i026           ;Se non ci sono altri caratteri da ricevere
        ifbs    Flag,FlErrorPresent,i027 ;Se c'è un messaggio di errore o informativo in coda
        ifeel   State,StStartOfLine,i027 ;E siamo nello stato StartOfLine
```

```
                                        ;(la riga precedente è stata processata)
        movf    Errore,W        ;Lo invia sulla seriale con la routine sendMsg
                call    sendMsg
                bcf     Flag,FlErrorPresent      ;Resetta il flag di errore presente
        ifend   i027
        goto    nextChar                ;Attende il carattere successivo
ifend   i026

[...]

; ------------------
; ----- errore -----
; ------------------
; Accoda il messaggio di errore passato in W in Errore (variabile globale)
; Appena cessa la ricezione di caratteri dalla seriale, verrà trasmesso l'errore generato
; Se è già presente un errore, viene conservato il primo e scartati gli ultimi

errore
        btfss   Flag,FlErrorPresent
        movwf   Errore
        bsf     Flag,FlErrorPresent
        return
```

Quando viene generato un errore, il relativo codice viene semplicemente messo nell’accumulatore e viene chiamata la routine errore, che setta il flag FlErrorePresent e si memorizza il codice di errore in una variabile globale chiamata Errore. Ulteriori errori non verranno memorizzati, poiché non è stato previsto un sistema di accodamento degli errori. L’errore verrà comunicato all’utente solo alla fine della ricezione della riga corrente, in modo da evitare perdita di byte sulla seriale, dato che non è stato implementato alcun controllo di flusso. In alto nel codice mostrato è presente lo stralcio del programma principale che si incarica di inviare i messaggi di errore.

Inizializzazione

Abbiamo già detto dell’importanza di una inizializzazione corretta, e mostrato qualche esempio. In questo paragrafo descriviamo riga per riga il codice di inizializzazione implementato nel firmware finale.

Il PIC comincia ad eseguire le istruzioni a partire da quella memorizzata all’indirizzo zero.

```
; ------------------------
; ----- Reset Vector -----
; ------------------------
; Il PIC inizia l'esecuzione delle istruzioni da qui

        ORG     0x0000  ; Reset Vector
        goto    init    ; Si saltano le definizioni di tavole in pagina zero
```

Nel nostro codice l’istruzione presente all’indirizzo zero è semplicemente una istruzione di salto incondizionato alla sezione init. La sezione init è spostata oltre la pagina zero poiché in pagina zero è opportuno memorizzare le look-up tables.

```
; ------------------------
; ----- Main Program -----
; ------------------------
; Sezione init eseguita una sola volta
; Sezione nextchar eseguita in ciclo ad eternum
```

```
init
        movlw   0x20            ;Cancella i file register da 0x20 a 0x7f
        movwf   FSR             ;in modo da assicurare che tutte le variabili
initL   clrf    INDF            ;del programma siano a zero inizialmente
        incf    FSR,F           ;
        btfss   FSR,7           ;
        goto    initL

        movlw   TrisPortA       ;Imposta i TRIS register per le due porte
        tris    PORTA           ;
        movlw   TrisPortB       ;
        tris    PORTB           ;
        movlw   B'00000111'     ;Configura PORTA come Digital I/O
        movwf   CMCON           ;xxxxx111 Comparator disable, RBA0-3 as Digital I/O
        clrf    PORTA           ;Clear port latches
        clrf    PORTB           ;

        bsf     STATUS,RP0      ;Bank 1
        movlw   B'00100110'     ;Configura USART
        movwf   TXSTA           ;Trasmit Status and Control: 8-bit receive,
                                ;Trasmit enable, Asincronous mode, Baud Rate Generator High
                Speed
        movlw   .20             ;Configura USART
        movwf   SPBRG           ;Baud Rate Generator Register  20: 59524 (57600) @ 20 MHz
        bcf     STATUS,RP0      ;Bank 0
        movlw   B'10010000'     ;Configura USART
        movwf   RCSTA           ;Receive Status and Control: Serial Port Enable,
                                ;8-bit receive, Continuous receive enable

        BuzzOff                 ;Spegne cicalino
        LedOff                  ;Spegne Led

        bcf     Flag,FlVerboseAutotest  ;Determina numero di moduli collegati,
                                        ;con verbose OFF (non stampa "1" o "0")
        call    autotest                ;
        bsf     Flag,FlVerboseAutotest  ;Verbose On per tutti gli usi successivi (*TST?)

        ifgtel  SlotN,.1,i001           ;Se sono stati trovati tra 1 e 4 moduli
        ifltl   SlotN,.5,i001
                movlw   0x02                    ;02 Hallo
                call    sendMsg                 ;
        ifend   i001                    ;Altrimenti all'accensione (o dopo *RST)
                                        ;la stringa di errore stampata da autotest

        call    aggiornaModuli          ;Tutti i relais aperti

nextChar
[...]
goto    nextChar
```

Le prime istruzioni eseguite sono quelle di inizializzazione: le variabili del programma vengono inizializzate con valori opportuni, i “gadget hardware” del PIC vengono configurati per l’uso desiderato, e si eseguono quindi procedure iniziali che vanno eseguite una sola volta, prima di passare il controllo al ciclo principale, che resterà in esecuzione fino allo spegnimento del circuito.

Per avere a disposizione PORTA come ingressi e uscite digitali, vengono disattivati i comparatori analogici, multiplexati sugli stessi pin, e attivi per default. Vengono impostati i registri di direzione delle porta (TRISA e TRISB) in funzione dei collegamenti hardware del PIC. Infine viene configurata la USART per una trasmissione seriale asincrona a 57600 baud, 8 bit, 1 bit di start, 1 bit di stop, senza l’uso di interrupt, e con trasmissione e ricezione continue, anziché attivate su richiesta. Per come sono collegati il led e il buzzer, questi sono attivi con una uscita bassa, occorre quindi spegnerli portando l’uscita a valore alto.

Infine viene eseguita la routine di autotest e determinato il numero di moduli collegati al controller, segnalando il buon esito dell'operazione con il messaggio *02 Hallo* oppure il fallimento con uno dei due messaggi di errore *09 Interrupted chain* oppure *15 Too many modules*.

Autotest

L'autotest invia un pattern di bit sugli shift-register, e verifica dopo quanti colpi di clock lo stesso pattern si ripresenta sul piedino di feedback. Il piedino di feedback è dotato, in hardware, di una resistenza di pull-up, così da fornire sempre 1 se non è stato collegato alcun modulo.

```
;--------------------
;----- autotest -----
;--------------------
; Invia pattern di bit sugli shift register (senza mai mandare store, che aggiornerebbe
; anche i Latch di uscita). Siccome i diversi moduli sono collegati in anello, con
; terminatore sull'ultimo, i bit dopo aver percorso l'anello tornano al microcontrollore
; dal piedino di Feedback. autotest verifica quanti moduli sono connessi, conservando
; l'informazione in SlotN, oppure segnala gli errori di catena interrotta
; o troppi moduli connessi,
; inviando direttamente l'errore sulla seriale, e settando SlotN a 0 o 5.
; Se Flag[FlVerboseAutotest] è settato, invia 1 o 0 sulla seriale a seconda
; se l'autotest ha successo o no

autotest
        clrf    SlotN

        bcf     ShRegPort,DataBit                ; Invia 0
        forl    autotestLooper,0xFF,i020 ; agli shift register
                bcf     ShRegPort,ClockBit       ;
                bsf     ShRegPort,ClockBit       ;
        next    autotestLooper,i020              ;
        bcf     ShRegPort,ClockBit               ; 257 volte
        bsf     ShRegPort,ClockBit               ;

        ifbs    ShRegPort,FeedbackBit,i022
                        ; Se trova 1 su Feedback (che ha un resistore di pull-up su scheda)
                call    autotestFallito
                movlw   0x09                     ; 09 Interrupted chain
                call    errore
                return
        ifend   i022

        forl    autotestLooper,.4,i021           ; Per 4 volte
                movlw   0xFF                     ; Invia 16 1 sugli shift register
                call    shiftout
                call    shiftout
                incf    SlotN,F
                ifbc    ShRegPort,FeedbackBit,autotestOk
                                ;Se trova 1, abbiamo determinato il numero di moduli connessi
        next    autotestLooper,i021              ; Altrimenti ci sono più moduli

        call    autotestFallito                  ; I moduli connessi sono più di 4
        movlw   0x15                             ; 15 Too many modules
        call    errore

        return

autotestOk
        ifbs    Flag,FlVerboseAutotest,i067
                movlw   "+"
                call    trasmit
                movlw   "1"
                call    trasmit
                call    newline
```

```
        ifend   i067
        return

autotestFallito
        ifbs    Flag,FlVerboseAutotest,i066
                movlw   "+"
                call    trasmit
                movlw   "0"
                call    trasmit
                call    newline
        ifend   i066
        return
```

La routine accetta un parametro booleano in ingresso: Flag[FlVerboseAutotest]. Se è settato sarà cura della stessa routine inviare un "+1" o "+0" sulla seriale in caso di successo o insuccesso del test (utile nel caso di risposta al comando *TST?). Se Flag[FlVerboseAutotest] non è settato, l'autotest sarà "silenzioso", e questo è utile durante l'inizializzazione del programma.

Ciclo principale

Il ciclo principale del programma è sostanzialmente costituito da:

- Eventuale comunicazione di messaggi di errore accodati
- Gestione degli errori della porta seriale
- Lettura del successivo carattere dalla seriale
- Condizionamento del carattere letto
- Fornitura del carattere al Parser per l'analisi dell'input
- Il Parser stesso chiama eventualmente la routine che esegue un comando riconosciuto.

Della comunicazione di errori accodati abbiamo già discusso, vediamo il resto:

```
nextChar

[...]

        ifbs    RCSTA,OERR,i072          ;Test for overrun error
                movlw   0x13             ;13 RS232 Overrun
                call    errore
                letl    State,StSeekNewLine
                bcf     RCSTA,CREN       ;Riinizializza la logica di ricezione disattivandola
                bsf     RCSTA,CREN       ;e riattivandola subito dopo; si è perso
                                         ;(almeno) un byte
        ifend   i072

        ifbs    RCSTA,FERR,i074          ;Test for framing error
                movlw   0x12             ;12 RS232 Framing
                call    errore
                movf    RCREG,W          ;Scarta il byte, perchè contiene dati non validi
        letl    State,StSeekNewLine      ;Stato: cerca inizio nuova linea
        ifend   i074

        movf    RCREG,W                  ;Legge il carattere successivo
        movwf   LastChar
        ifltl   LastChar,'z'+1,i005      ;Converte in maiuscole le minuscole "a" .. "z"
        ifgtel  LastChar,'a',i005
                movlw   'A'-'a'
                addwf   LastChar,F
        ifend   i005
        ifeel   LastChar,0x03,i109       ;CTRL-C : Break
                goto    nextChar         ;Semplicemente ignorato, per compatibilità con SCPI
```

```
        ifend   i109

StateCase                       ;Tavola di salto per stato corrente
        bsf     PCLATH,0        ;La macchina a stati del parser comincia da qui e processerà
        bcf     PCLATH,1        ;il carattere in LastChar (ultimo letto dalla seriale)
        movf    State,W
        addwf   PCL,F
        goto    stStartOfLine
        goto    stSeekNewLine
        goto    stParseId
        goto    stNewLineAfterQm
        goto    stChiocciolaOnly
        goto    stRiempiLista
        goto    stNewLineAfterParClose
        goto    nextChar ; Should Never come here

        IF ((HIGH ($)) != (HIGH (StateCase)))
                ERROR "La tavola di salto StateCase ha superato i confini di pagina"
        ENDIF
        IF ((HIGH ($)) != B'01')
                ERROR "La tavola di salto StateCase ha bisogno di ridefinire PCLATH"
        ENDIF
```

In caso di errore di OverRun (possibile solo se il PC invia due comandi di seguito senza leggere prima la risposta del microcontrollore, poiché l'unico processo bloccante sul PIC è l'invio di caratteri), oltre a comunicare l'errore (che genererà una cascata di errori di overrun, uno ogni 100 ms, fino a che il trasmittente non fa una pausa), occorre fermare e far ripartire la logica di ricezione della USART, come specificato sul datasheet del PIC.

In caso di errore di trama, conviene scartare il byte letto poiché contiene dati non validi.

Il carattere letto dalla seriale viene memorizzato nella variabile globale LastChar.

Il condizionamento del carattere consiste nel convertire le lettere minuscole in maiuscole, poiché SCPI è un protocollo case-insensitive.

La combinazione CTRL-C, prevista dallo standard SCPI, deve interrompere elaborazioni lunghe nella macchina, riportandola in uno stato in cui presta attenzione al PC. Siccome non esistono elaborazioni lunghe in questo progetto, il carattere CTRL-C viene semplicemente accettato e scartato, per mantenere la compatibilità con lo standard. E' più semplice eliminarlo semplicemente dal flusso di dati piuttosto che passarlo al parser per l'analisi.

Lo stato principale della macchina a stati del parser è memorizzato nella variabile globale State, che assume valori pari alle costanti Stxxxx, dichiarate all'inizio del programma:

```
        cblock  0x00
;--- Valori possibili per State (stato corrente della macchina a stati di parsing)
StStartOfLine           ; Programma appena avviato, oppure riga precedente terminata
                        ; e inizio riga successiva
                        ; Cerca "*" o l'inizio di un identificatore ("A".."Z")
StSeekNewLine           ; La linea precedente ha generato un errore, oppure non interessa
                        ; leggerne la fine, si attende la conclusione della riga attuale
                        ; senza processare nulla. Il led rimane acceso in questo stato.
StParseId               ; Si riempie Ident con l'identificatore man mano che ne arrivano
                        ; i caratteri costituenti. L'identificatore verrà riconosciuto
                        ; non appena si ricevono caratteri consecutivi adeguati.
                        ; Una volta riconosciuto l'identificatore, si setta il SubSystem
                        ; di default e il SubSystem di cui si è appena fatto il parsing.
                        ; Un comando correttamente riconosciuto non verrà eseguito subito,
                        ; ma a fine riga. Si trattano in questo stato anche i duepunti ":",
                        ; spazi " ", parentesi aperta "(", punto interrogativo "?" e
                        ; il carattere newline che termina il comando corrente
                        ; che verrà eseguito se riconosciuto.
StNewLineAfterQm        ; E' stata riconosciuta una query, dopo il ? si attende newline
```

```
                         ; Ricevuto il newline, verrà eseguito il comando
                         ; e inviati i risultati
StChiocciolaOnly         ; E' stata riconosciuta la parentesi aperta dopo un route:open
                         ; o route:close, il prossimo carattere deve essere per forza @
StRiempiLista            ; Parsing della lista di canali da chiudere/aprire.
                         ; Si attendono ulteriori numeri di canale, virgole "," o la ")"
StNewLineAfterParClose   ; E' stato riconosciuto un ROUTe:OPEN (@ ...) o ROUTe:CLOSe (@ ... )
                         ; Dopo la ) si attende per forza newline. Ricevuto il newline,
                         ; si esegue il comando
        endc
```

Il passaggio del controllo di flusso allo stato corretto del Parser viene eseguito mediante una tavola di salto analoga all’istruzione on ... goto del Basic. Le etichette utilizzate per individuare le sezioni di codice che implementano i diversi stati del parser hanno lo stesso nome delle costanti appena descritte, ma l’iniziale minuscola.

Caratteristiche utili per debug e soluzioni poco eleganti ma efficaci

Nel ciclo principale sono stati inserite altre funzioni, un po’ poco eleganti ma molto efficaci.

La prima è l’accensione del Led quando il parser si trova nello stato stSeekNewLine: una programmazione più elegante doveva scrivere l’istruzione di accensione Led contestualmente al cambio di stato, ogni volta che questo si presenta: effettuando il controllo nel ciclo principale, si è risparmiata però memoria, a vantaggio dell’efficienza e della mantenibilità del codice (se occorre effettuare altre operazioni, oltre ad accendere il led, se ne può inserire il relativo codice una volta sola).

```
nextChar
        ifeel   State,StSeekNewLine,i010 ; Il led viene acceso quando
                                         ; si è nello stato StSeekNewLine
                LedOn                    ;    Viene    spento    direttamente    dallo    stato
StSeekNewLine
                                         ; non appena si riceve newline
        ifend   i010

[...]

        ifeel   LastChar,0x0D,i101       ; CTRL-M : Carriage Return
                goto    nextChar         ; Semplicemente ignorato, così da poter riconoscere
                                         ; CR+LF come terminatore di riga valido
        ifend   i101
```

Il carattere CTRL-M (0x0D, Carriage Return) viene semplicemente ignorato, così come veniva ignorato CTRL-C, così da considerare egualmente valido, come terminatore di riga, la coppia CR/LF anziché LF da solo. Lo standard SCPI indica solo LF come terminatore di riga valido, si è voluto aggiungere il supporto per CR/LF in modo da facilitare l’interazione con la macchina tramite il terminale di Windows, che per impostazione predefinita invia la coppia CR/LF quando si preme INVIO. Nessun controllo viene eseguito sulla posizione di CTRL-M, che può anche non essere adiacente al LF senza che la macchina riporti errori. Non è stato ritenuto necessario identificare questa situazione come errore, trattandosi di una caratteristica aggiuntiva, non prevista dallo standard, e implementata solo perché utile al debug. Sul

prodotto finale, questa caratteristica si può benissimo eliminare, pur restando compatibili con lo standard.

Dello stesso tenore è l'implementazione di due funzioni fuori-standard, attivate dai caratteri CTRL-E e CTRL-F: il primo attiva l'echo dei caratteri ricevuti, e il secondo attiva l'invio di terminatori di riga con la coppia CR/LF anziché con LF soltanto. Anche queste due funzioni sono molto utili nell'interazione con la macchina tramite il terminale di Windows, ma andrebbero eliminate dalla versione finale per compatibilità piena con lo standard SCPI, oppure lasciate, come estensioni proprietarie dello standard. L'implementazione è poco elegante poiché si utilizza lo stesso carattere per attivare e disattivare la funzione: una implementazione più elegante dovrebbe poter prevedere la possibilità di indicare esplicitamente l'attivazione o disattivazione di una funzione. Si è fatta questa scelta poiché la finalità è il debug in interattiva al terminale di Windows, e l'operatore umano è in grado di riconoscere immediatamente se la funzione è attiva o disattiva e regolarsi di conseguenza.

```
ifeel   LastChar,0x05,i096       ;Ricevuto CTRL-E (combinazione fuori-standard)
        negbit  Flag,FlEcho      ;Attiva o disattiva l'echo dei caratteri ricevuti
        goto    nextChar         ;Utile per il debug al terminale di Windows
ifend   i096

ifeel   LastChar,0x06,i100       ;Ricevuto CTRL-F (combinazione fuori-standard)
        negbit  Flag,FlCrLf      ;Attiva o disattiva il CR accanto al LF in tx
        goto    nextChar         ;Utile per il debug al terminale di Windows
ifend   i100

ifbs    Flag,FlEcho,i097         ;Se è attivo l'echo dei caratteri ricevuti
movf    LastChar,W               ;viene reinviato al PC ogni carattere ricevuto
        call    trasmit
ifend   i097
```

Parser

Accendere e spegnere dei relais collegati a shift-register è tutto sommato piuttosto facile con un microcontrollore, mentre realizzare il parser, ovvero quello strumento software in grado di riconoscere comandi impartiti tramite stringhe di testo, e che restituisce gli errori di sintassi non valida, è tutt'altro che banale, specie in assembler.

Non è la prima volta che realizzo un parser per riconoscere un linguaggio con pochi costrutti[58], però è la prima volta che implemento queste funzionalità in un linguaggio a basso livello.

[58] Ho realizzato infatti un parser in Java, a scopo didattico, in grado di riconoscere i costrutti XML. Si veda il mio tema d'anno del corso di Sistemi Informativi, dal titolo "Generazione dell'albero corrispondente ad un documento XML mediante un parser scritto in Java", al quale rimando per chi vuole approfondire la conoscenza degli algoritmi di parsing, e del mio personalissimo stile nell'implementarli.

Il parser è implementato come macchina a stati. Un'attenta progettazione ha consentito di mantenere basso il numero di stati presenti nella macchina, che ammontano a 7 in tutto[59]. Una descrizione testuale degli stati presenti è stata già illustrata mostrando il ciclo principale del programma. La descrizione degli stati era presente come commenti del codice. Ora diamo una rappresentazione grafica della macchina a stati.

Diagramma degli stati del parser

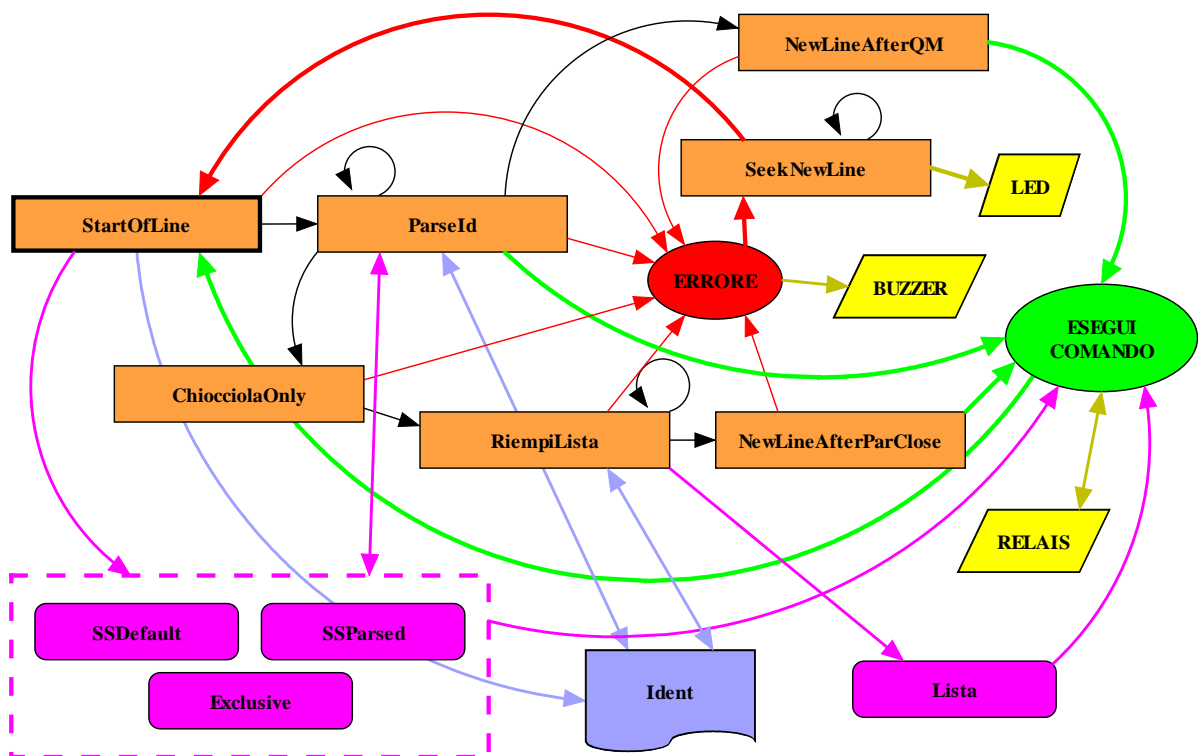



Diagramma degli stati del parser

Nel diagramma, in arancione sono mostrati gli stati principali (in bordo ingrossato lo stato di inizio, nel quale si torna per ogni nuova riga), in nero i percorsi di cambio di stato, in verde i percorsi che portano al riconoscimento di un comando valido e alla sua esecuzione, in rosso i percorsi che portano alla generazione di messaggi di errore, in giallo gli accessi alle sottoroutine del Bios, e da questi all'hardware. In azzurro la variabile di tipo stringa che conserva i successivi caratteri di un identificatore in riconoscimento al momento, e in fucsia altre variabili di stato accessorie che conservano la storia del recente passato della macchina a

[59] Procedendo "alla cieca" nello scrivere la macchina a stati, ci si ritroverebbe invece con un centinaio di stati differenti, con un percorso tra essi per ciascun comando da riconoscere, e scarse o nessuna possibilità di mantenimento del codice e aggiunta di nuove funzionalità senza rivedere pesantemente la struttura degli stati.

stati, e memorizzano in una struttura di memoria compatta i costrutti della sintassi già riconosciuti.
I compiti demandati a ciascuno stato sono:

- Processare un singolo carattere del comando in arrivo, che il ciclo principale del programma ha memorizzato nella variabile globale LastChar;
- Richiamare eventualmente il sottosistema di generazione e accodamento messaggio di errore (l'errore verrà memorizzato nella coda e comunicato dal ciclo principale al termine della riga corrente);
- Eseguire eventualmente un comando valido ricevuto dopo la ricezione di new-line, che termina correttamente il comando;
- Settare la variabile State sullo stato seguente opportuno (eventualmente anche sé stesso, se il successivo carattere atteso va processato dallo stesso stato);
- Restituire il controllo al ciclo del programma principale con l'istruzione goto nextChar.

I valori attuali assunti delle variabili accessorie, rappresentate in fucsia e azzurro sul diagramma, sono in grado di modificare il comportamento della macchina, che esibirà un comportamento differente in base a:

- Ultimo carattere letto;
- Stato principale corrente;
- Quanto è stato memorizzato, nel recente passato, nelle variabili di stato accessorie.

Quindi il numero degli stati ammonta a 7 soltanto, ma ciascuno di questi esibisce un comportamento complesso, differente in base alla storia del recente passato, memorizzata nelle variabili accessorie, ed è quindi come se il parser fosse costituito da molti più stati differenti.
L'esecuzione del comando, o la segnalazione di un messaggio di errore, viene ritardata fino al termine della riga, senza che questo comporti la necessità di bufferizzare tutta la riga di testo come sequenza di caratteri: di quanto già ricevuto e riconosciuto, si tiene traccia nelle variabili di stato accessorie, che mantengono una rappresentazione molto compatta in memoria dei costrutti già riconosciuti.

Variabili di stato accessorie

Queste conservano:

- **SSDefault (1 byte)**: codice del subsystem di default, già settato da comandi precedenti, o appena settato durante l'analisi della riga di comando attuale;

- **SSParsed (1 byte)**: codice dell'identificatore appena riconosciuto;
- **Exclusive (1 bit)**: settato se si è riconosciuto l'identificatore EXCLUSIVE, che è l'unico identificatore di terzo livello esistente nella sintassi: se ce ne fossero di più, si dovrebbe sostituire questa variabile con un'altra da un byte in grado di memorizzare più codici differenti[60];
- **Lista (8 byte):** utilizzata solo durante il parsing di channel list (comandi ROUTE:CLOSE e ROUTE:OPEN), memorizza in maniera molto compatta (1 bit per ogni relais dei 64 possibili su 4 moduli, per un totale di 8 byte) la lista di canali ricevuta, in modo da consentire il posticipamento, ma soprattutto l'esecuzione in contemporanea dei comandi di apertura/chiusura di relais. L'alternativa, non praticabile, avendo deciso di non bufferizzare l'intera stringa di comando ricevuta, era di aprire/chiudere i relais man mano che si riceveva il relativo numero di canale valido. Così facendo, però, ci si sarebbe ritrovati nell'impossibilità di abortire, evitandone l'esecuzione, un comando che conteneva un errore di sintassi in coda;
- **Ident:** variabile stringa a lunghezza dinamica, implementata con routine di gestione stringa, che contiene i successivi caratteri dell'identificatore, o numero di canale, in riconoscimento. Man mano che un identificatore viene riconosciuto, Ident viene svuotata e quindi sovrascritta dall'identificatore successivo, dopo aver memorizzato adeguatamente nelle variabili di stato accessorie l'identificatore riconosciuto.

Le dichiarazioni di queste variabili, e dei codici numerici che possono assumere, sono fatte all'inizio del programma:

```
State            ; Stato della macchina a stati di parsing
SSDefault        ; Ultimo subsystem utilizzato, che resta come default
SSParsed         ; SubSystem il cui identificatore è in parsing.
                 ;  Inizialmente vale SSUndefined per permettere di riconoscere header non
validi

[...]

FlExclusive      equ      5
        ; Settato se è riconosciuto il tag EXCLusive dopo ROUTe:CLOSe
        ; siccome è l'unico tag di 3° livello implementato,
        ; si è preferito un Flag di un bit anziché una apposita variabile
        ; di SubSystem di 3° livello riconosciuto

[...]
```

---

[60] Inizialmente si pensava di mantenere il codice più generico e utilizzare 3 variabili: SS1, SS2, SS3 per i codici dei subsystem di primo, secondo, terzo livello, e una variabile Nest che indicasse il livello al quale fosse giunto il parsing attuale. Ciascuna di queste tre variabili andava poi raddoppiata per riconoscere un identificatore non valido nel parsing attuale ma conservare comunque informazione sul subsystem di default, settato nei parsing precedenti. Con una analisi più attenta del set di comandi da riconoscere, si è poi riusciti a semplificare il parser. Ecco un altro esempio di progettazione più laboriosa del programma per ottenere una codifica più semplice.

```
        cblock  0x00
;--- Valori possibili per SSDefault e SSParsed
SSUndefined     ; Valore predefinito iniziale.
                ; Se non viene modificato, consente di rilevare un header non riconosciuto
SSCommon        ; Common Commands (Livello 1)
SSSystem        ; System (Livello 1)
SSRoute         ; Route (Livello 1)
SSMemory        ; Memory (Livello 1)
SSClose         ; Close (Livello 2, sotto /Route)
SSOpen          ; Open (Livello 2, sotto /Route)
SSExclusive     ; Exclusive (Livello 3, sotto /Route/Close)
SSCpon          ; Cpon (Livello 2, sotto /System)
SSVer           ; Version (Livello 2, sotto /System)
SSModule        ; Module (Livello 2, sotto /System)
SSState         ; State (Livello 2, sotto /Memory)
SSDelete        ; Delete (Livello 3, sotto /Memory/State)
SSValid         ; Valid (Livello 3, sotto /Memory/State)
SSIdn           ; Idn (Livello 2, sotto /Common)
SSTst           ; Tst (Livello 2, sotto /Common)
SSRst           ; Rst (Livello 2, sotto /Common)
SSSav           ; Sav (Livello 2, sotto /Common)
SSRcl           ; Rcl (Livello 2, sotto /Common)
SSCls           ; Cls (Livello 2, sotto /Common)
        endc
```

Operazioni demandate a ciascuno stato

Segue il commento del codice che implementa ciascuno stato, con la descrizione delle funzionalità svolte.

StartOfLine

E' lo stato di inizio, al quale si torna ad ogni nuova riga. Il set di comandi da riconoscere accetta come carattere di inizio solo un **asterisco** o il primo **carattere alfabetico** di un identificatore. Nel primo caso si setta il subsystem di default a quello dei common commands, nel secondo caso si inserisce il primo carattere dell'identificatore in parsing, in accumulo nella stringa ident. **Ogni altro carattere** deve dar luogo all'errore di sintassi *05 Invalid Char*. In caso di errore, lo stato successivo è *SeekNewLine* (al quale ogni causa di errore porta), in caso di non errore lo stato successivo è *ParseId* (osservare il diagramma degli stati per seguire questi ragionamenti).

Se si riceve un **new-line** proprio a inizio riga, è segno che si è ricevuta una riga vuota, e si permane in questo stato, non eseguendo alcuna operazione. Si è deciso di non rispondere con *01 Completed* al comando consistente in una riga vuota, ma di non svolgere semplicemente alcuna operazione.

Quanto descritto fin'ora è sufficiente al riconoscimento della sintassi dei comandi. Si è deciso poi di estendere appena le funzionalità di StartOfLine accettando il carattere di **":" (due punti)** ad inizio riga, condizione nella quale si permane in StartOfLine, poiché questo è il comportamento di molte macchine programmabili da laboratorio, anche se non documentato. Così come è scritto, il parser accetta anche più caratteri di due punti consecutivi. Volendo

segnalare un errore in queste condizioni, occorreva aggiungere un contatore del numero dei due punti ad inizio riga come variabile di stato accessoria, ma si è deciso di non effettuare questo controllo di errore ulteriore, che avrebbe appesantito inutilmente il codice, poiché un tale comportamento della macchina non è fonte di errori critici. Vale invece la pena di spendere del tempo in controlli di sintassi ulteriori quando una sintassi male interpretata può produrre comportamenti non desiderati della macchina.

```
stStartOfLine
        clrf    SSParsed         ; All'inizio di una nuova riga, si cancellano i SubSystem
        bcf     Flag,FlExclusive  ; riconosciuti precedentemente, a parte il SubSystem di
        clrf    IdentLength      ; default, che rimane valido
        ifeel   LastChar,"*",i002        ;Incontrato "*": Subsystem dei common commands,
                                         ;si processa l'header seguente
                letl    SSDefault,SSCommon
                letl    State,StParseId
                goto    nextChar
        ifend   i002

        ifgtel  LastChar,"A",i004        ;Incontrato un carattere alfabetico, si inizia a
                                         ;riempire Ident con l'header in riconoscimento
        ifltl   LastChar,"Z"+1,i004
                movf    LastChar,W
                call    IdentAddChar
                letl    State,StParseId
                goto    nextChar
        ifend   i004

        ifeel   LastChar,":",i011        ;Incontrato ":" ad inizio riga: ignorato
                goto    nextChar
        ifend   i011

        ifeel   LastChar,0x0A,i014       ;Incontrato newline ad inizio riga (riga vuota)
                                         ;semplicemente ignorato
                goto    nextChar
        ifend   i014

        movlw   0x05                     ;Incontrato un carattere diverso: 05 Invalid Char
        call    errore
        letl    State,StSeekNewLine
        goto    nextChar
```

ParseId

E' lo stato dal codice più complesso, in quanto è demandato a questo stato il riconoscimento degli identificatori e di tantissimi costrutti sintattici, o almeno del loro inizio (poi sono stati successivi che continuano l'opera di riconoscimento di sintassi specifiche).

Nel funzionamento di base, ParseId aggiunge i caratteri alfabetici ricevuti a Ident, fino a riconoscere un identificatore valido, che viene memorizzato in SSParsed. Se il SubSystem riconosciuto è uno di quelli che devono permanere come SubSystem di default, si memorizza tale SubSystem anche in SSDefault.

E' così piuttosto semplice aggiungere il supporto per nuovi comandi, semplicemente inserendo i caratteri costitutivi dell'identificatore da riconoscere e memorizzando in SSParsed il comando riconosciuto, senza cambiare minimamente la struttura della macchina a stati.

Se si riceve **new-line**, è possibile che si sia ricevuto un intero comando valido, che va eseguito (per esempio *RST): in questo stato sono presenti infatti le routine di esecuzione dei comandi con sintassi più semplice.

Se si ricevono i **“:” (due punti)**, si cancella Ident per far posto al riconoscimento dell’identificatore successivo. Osservare il codice per la gestione dell’identificatore EXCLUSIVE, l’unico caso di identificatore di terzo livello presente.

Se si riceve **“ ” (uno spazio)**, ci si trova di fronte ad un comando con parametro (per esempio SYSTEM:CPON xxx) e occorre cancellare Ident per memorizzarvi dentro il parametro. Il comando SYSTEM:CPON viene gestito integralmente da questo stato.

Se si riceve **“(” (Parentesi aperta)**, il carattere è accettato solo come inizio di lista di canali dopo un ROUTE:CLOSE o ROUTE:OPEN, altrimenti si genera un errore *05 Invalid Char*. Il controllo per lo stato successivo viene passato a *StChiocciolaOnly*, il primo della catena di 3 stati che decodifica le liste di canali.

Se si riceve un **“?” (Punto interrogativo)**, il flusso per lo stato successivo passa a *StNewLineAfterQM*, lo stato che esegue una query dopo aver ricevuto Newline.

Come sempre, in caso di errori, dopo aver invocato il gestore di errori, lo stato successivo viene impostato a *StSeekNewLine*. Per evitare di appesantire la trattazione, non ho citato tutte le fonti possibili di errore: osservare direttamente i commenti nel codice per capire come sono state trattate.

```
stParseId
        ifgtel  LastChar,"A",i007        ; if LastChar in ["A".."Z"] U ["0".."9"]
        ifltl   LastChar,"Z"+1,i007
                goto    alfanum
        ifend   i007
        ifgtel  LastChar,"0",i017        ; 0..9 processati per SYSTEM:CPON x00
        ifltl   LastChar,"9"+1,i017
                goto    alfanum
        ifend   i017
        ifelse  alfanum,notalfanum
                ;Lo stato ParseId, in caso di caratteri alfanumerici, aggiunge caratteri a
                ;Ident fino a riconoscere un identificatore. Quando un header è riconosciuto,
                ; si settano le variabili SSDefault, SSParsed, Flag(FlExclusive)
                ; opportunamente in base a quanto si è riconosciuto. Se si riconosce un
                ; identificatore non appartenente al SubSystem corrente, si ignora
                ; semplicemente il riconoscimento: l’errore verrà segnalato alla fine della
                ; riga corrente poiché SSParsed, non aggiornata, è rimasta al suo valore
                ; iniziale SSUndefined
                movf    LastChar,W
                call    IdentAddChar

                ifident4 "R","O","U","T",i028
                        letl    SSParsed,SSRoute
                        letl    SSDefault,SSRoute
                ifend   i028

                ifident4 "C","L","O","S",i029
                ifeel   SSDefault,SSRoute,i029
                        letl    SSParsed,SSClose
                ifend   i029

                ifident4 "O","P","E","N",i030
                ifeel   SSDefault,SSRoute,i030
                        letl    SSParsed,SSOpen
                ifend   i030
```

```
        ifident4 "E","X","C","L",i078
        ifeel   SSDefault,SSRoute,i078
                ifeel   SSParsed,SSClose,i079
                        bsf     Flag,FlExclusive
                        goto    nextChar
                ifend   i079
                ifeel   SSParsed,SSOpen,i080
                        bsf     Flag,FlExclusive
                        goto    nextChar
                ifend   i080
        ifend   i078

        ifident4 "S","Y","S","T",i033
                letl    SSParsed,SSSystem
                letl    SSDefault,SSSystem
                goto    nextChar
        ifend   i033

        ifident4 "M","O","D","U",i076
        ifeel   SSDefault,SSSystem,i076
                letl    SSParsed,SSModule
                goto    nextChar
        ifend   i076

        ifident4 "C","P","O","N",i036
        ifeel   SSDefault,SSSystem,i036
                letl    SSParsed,SSCpon
                goto    nextChar
        ifend   i036

        ifident3 "V","E","R",i035
        ifeel   SSDefault,SSSystem,i035
                letl    SSParsed,SSVer
                goto    nextChar
        ifend   i035

        ifident3 "I","D","N",i013
        ifeel   SSDefault,SSCommon,i013
                letl    SSParsed,SSIdn
                goto    nextChar
        ifend   i013

        ifident3 "T","S","T",i015
        ifeel   SSDefault,SSCommon,i015
                letl    SSParsed,SSTst
                goto    nextChar
        ifend   i015

        ifident3 "R","S","T",i023
        ifeel   SSDefault,SSCommon,i023
                letl    SSParsed,SSRst
                goto    nextChar
        ifend   i023

        ifident3 "C","L","S",i031
        ifeel   SSDefault,SSCommon,i031
                letl    SSParsed,SSCls
                goto    nextChar
        ifend   i031

        ifeel   SSDefault,SSSystem,i042
        ifeel   SSParsed,SSCpon,i042
; Riconosciuti gli header, si setta opportunamente SSParsed.
; Un comando riconosciuto non viene eseguito subito, ma si rimanda l’esecuzione
; a fine riga, così in caso di errori di sintassi successivi si può “disfare”
; l’intera riga ricevuta senza eseguire nulla. SYSTEM:CPON è l’eccezione a questa
; regola, e viene in parte eseguito direttamente, per evitare di complicare la
; macchina con altri stati e altre variabili temporanee: SYSTEM:CPON viene gestito
; integralmente da StParseId anche se concettualmente avrebbe bisogno di uno stato
; a parte, essendo un comando con parametro.
                ifident3 "A","L","L",i037
                        clrf    Relais
                        clrf    Relais+1
                        clrf    Relais+2
                        clrf    Relais+3
                        clrf    Relais+4
                        clrf    Relais+5
```

```
                        clrf    Relais+6
                        clrf    Relais+7
                        goto    nextChar
                ifend   i037
                ifeel   IdentLength,.3,i106
                        ifeel   Ident+1,"0",i044
                        ifeel   Ident+2,"0",i044
                                ifeel   Ident+0,"1",i046
                                        clrf    Relais
                                        clrf    Relais+1
                                        goto    nextChar
                                ifend   i046
                                ifeel   Ident+0,"2",i047
                                        ifltl   SlotN,.2,i071
                                                movlw   0x08
                                                ;08 No module in slot
                                                call    errore
                                                letl    State,StSeekNewLine
                                                goto    nextChar
                                        ifend   i071
                                        clrf    Relais+2
                                        clrf    Relais+3
                                        goto    nextChar
                                ifend   i047
                                ifeel   Ident+0,"3",i048
                                        ifltl   SlotN,.3,i073
                                                movlw   0x08
                                                ;08 No module in slot
                                                call    errore
                                                letl    State,StSeekNewLine
                                                goto    nextChar
                                        ifend   i073
                                        clrf    Relais+4
                                        clrf    Relais+5
                                        goto    nextChar
                                ifend   i048
                                ifeel   Ident+0,"4",i049
                                        ifltl   SlotN,.4,i075
                                                movlw   0x08
                                                ;08 No module in slot
                                                call    errore
                                                letl    State,StSeekNewLine
                                                goto    nextChar
                                        ifend   i075
                                        clrf    Relais+6
                                        clrf    Relais+7
                                        goto    nextChar
                                ifend   i049
                                ifgtel  Ident+0,"5",i102
                                        movlw   0x06    ;06 Slot out of range
                                        call    errore
                                        letl    State,StSeekNewLine
                                        goto    nextChar
                                ifend   i102
                                ifltl   Ident+0,"1",i105
                                        movlw   0x05    ;04 Unknown Header
                                        call    errore
                                        letl    State,StSeekNewLine
                                        goto    nextChar
                                ifend   i105
                        ifend   i044
                        movlw   0x04    ;04 Unknown Header
                        call    errore
                        letl    State,StSeekNewLine
                        goto    nextChar
                ifend   i106
        ifend   i042

        goto    nextChar
ifend   notalfanum

ifeel   LastChar,0x0A,i016
; Incontrato un newline, occorre eseguire il comando riconosciuto
        ifeel   SSDefault,SSCommon,i024
        ifeel   SSParsed,SSRst,i024
                goto 0
                ; *RST resetta il PIC rieseguendo l'inizializzazione
        ifend   i024
```

```
        ifeel   SSDefault,SSCommon,i032
        ifeel   SSParsed,SSCls,i032
                ;do nothing  ; *CLS viene riconosciuto ma non sortisce alcun effetto
        ifend   i032

        ifeel   SSDefault,SSSystem,i053
        ifeel   SSParsed,SSCpon,i053
                call    aggiornaModuli
                ; SYSTEM:CPON richiede di aggiornare gli shift-register
                ; i nuovi valori sono già stati memorizzati in Relais
        ifend   i053

        ifeel   SSParsed,SSUndefined,i088
                ; Se SSParsed è ancora SSUndefined, messaggio di errore
                movlw   0x04                    ; 04 Undefined Header
                call    errore
                letl    State,StStartOfLine
                goto    nextChar
        ifend   i088

        movlw   0x01    ;01 Completed   ; Parte comune all'esecuzione dei comandi:
        call    errore                  ; si visualizza 01 Completed
        letl    State,StStartOfLine     ; Stato successivo = StStartOfLine
        goto    nextChar
ifend   i016

ifeel   LastChar," ",i008
        ; Dopo uno spazio si attende l'inizio di un nuovo identificatore
        clrf    IdentLength
        goto nextChar
ifend   i008

ifeel   LastChar,"(",i050
        ; Parentesi aperta è carattere valido se in precedenza si è avuto
        bcf     Flag,FlLogic    ; ROUTE:CLOSE o ROUTE:OPEN (e si aspetta la @)
        ifeel   SSDefault,SSRoute,i051
                ifeel   SSParsed,SSClose,i052
                        bsf     Flag,FlLogic
                ifend   i052
                ifeel   SSParsed,SSOpen,i056
                        bsf     Flag,FlLogic
                ifend   i056
        ifend   i051
        ifbc    Flag,FlLogic,i054               ; Altrimenti messaggio di errore
                movlw   0x05                    ; 05 Invalid Char
                call    errore
                letl    State,StSeekNewLine
                goto    nextChar
        ifelse  i054,i055
                letl    State,StChiocciolaOnly
                goto    nextChar
        ifend   i055
ifend   i050

ifeel   LastChar,":",i009
                ; I due punti separano un header dal successivo
        clrf    IdentLength
                ; occorre pulire Ident e SSParsed per il nuovo header
        ifnel   SSParsed,SSClose,nextChar
        ; Se SSParsed è SSClose, non cancella SSParsed, poiché si può attendere
        ; ancora EXCLUSIVE. La condizione if-not-equal-literal va qui
        ; letta goto-if-equal-literal
        clrf    SSParsed
        goto    nextChar
ifend   i009

ifeel   LastChar,"?",i038
        ; Dopo un "?" si attende solo NewLine prima di eseguire la query
        letl    State,StNewLineAfterQm
        goto    nextChar
ifend   i038

movlw   0x05                    ; Nessuno dei caratteri precedenti: errore
call    errore                  ; 05 Invalid Char
letl    State,StSeekNewLine
goto    nextChar
```

Notare i frequenti comandi goto nextChar: quando un’elaborazione è finita, si restituisce il controllo al ciclo del programma principale. Notare anche che SYSTEM:CPON viene integralmente gestito da questo stato, con qualche trucco di programmazione.
Siccome è l’unico comando con parametro dopo uno spazio presente, la cosa può anche andare bene, poiché semplifica la macchina a stati. Se si desidera implementare in revisioni successive del firmware ulteriori comandi con parametro, è una buona idea dedicare uno stato separato al parsing del parametro, e aumentare il numero di variabili ausiliarie in modo da differenziare tra le varie situazioni con un codice più chiaro.

SeekNewLine

E’ lo stato in cui si arriva in condizione di errore. Il led permane acceso, in questo stato, ad avvisare che la macchina non sta più processando l’input, poiché si è verificato un errore, ma attende semplicemente il carattere new-line per segnalare l’errore ricevuto e passare all’elaborazione della riga successiva. Il Led viene gestito parte dal ciclo principale (che l’accende) e parte da questo stato (che lo spegne).

```
stSeekNewLine
        ; Si arriva in questo stato se c'è stato un errore nell'istruzione. Non si processa
        ; più nulla della riga attuale, e si attende solo il carattere new-line per iniziare
        ; il parsing della nuova riga. Il led rimane acceso quando si permane in questo stato,
        ; e spento quando vi si esce, in modo da avvisare l'operatore che la macchina a stati
        ; non sta processando nulla ma sta solo aspettando il carattere newline.
        ifeel   LastChar,0x0A,i003
                letl    State,StStartOfLine
                LedOff
        ifend   i003
        goto    nextChar
```

ChiocciolaOnly

Se *StSeekNewLine* è lo stato dal codice più semplice in assoluto, *StChiocciolaOnly* non è da meno: dopo la parentesi aperta relativa all’inizio di una lista di canali, la sintassi prescrive la presenza di una chiocciola. Questo stato verifica solo questo e passa il controllo a *StRiempiLista*, dopo avergli preparato variabili ausiliarie svuotate. *StRiempiLista* è il secondo stato della catena di tre stati che processa le liste di canali.

```
stChiocciolaOnly
        ifeel   LastChar,"@",i057       ; In questo stato si attende solo la @ dopo una (
        clrf    Lista                   ; e si azzerano le strutture di memoria
                clrf    Lista+1         ; che ospiteranno la lista di canali.
                clrf    Lista+2
                clrf    Lista+3
                clrf    Lista+4
                clrf    Lista+5
                clrf    Lista+6
                clrf    Lista+7
                clrf    IdentLength
                letl    State,StRiempiLista
        ifelse  i057,i070               ; Se non si riceve una @, messaggio di errore
                movlw   0x05            ; 05 Invalid Char
                call    errore
                letl    State,StSeekNewLine
```

```
        ifend   i070
        goto    nextChar
```

RiempiLista

In questo stato si processano i **caratteri numerici** costituenti i numeri di canale delle liste, e il carattere **“,” (virgola)** che separa un numero di canale dal successivo. Viene inoltre riconosciuto il carattere **“)” (parentesi chiusa)** che termina la lista di canali. In questo caso il controllo passa al terzo e ultimo stato della catena di tre stati che riconosce le liste di canali, *StNewLineAfterParClose*.

Numerosi controlli per errori di sintassi, e anche per canali o slot fuori-range massimo o non fisicamente presenti con il numero di moduli collegati, vengono trattati in questo stato. Per ciascuno di essi si procede al solito modo: si setta l’errore tramite la chiamata al sottosistema di gestione degli errori, si impone come stato successivo *StSeekNewLine*, si passa il controllo al ciclo principale del programma con goto nextChar.

Le liste di canali dichiarate per intervalli (con il simbolo di due punti) non vengono implementate, ma comunque riconosciute, in modo da lasciare un *place holder* nel codice, per sviluppi futuri: volendo scrivere il codice per implementare le liste di canali, basta arricchire il *place holder*, già sistemato nel posto giusto nel codice.

```
stRiempiLista
        ; In questo stato si riempie la variabile Lista man mano che vengono ricevuti numeri
di
        ; Slot e Canale validi. Numerosi controlli per errori vengono rilevati, che faranno
        ; abortire l'intero comando. Per questo non si settano direttamente i bit di Relais,
        ; ma si prepara a parte la Lista di canali e si rimanda alla fine della riga
        ; l'esecuzione dell'aggiornamento.
        ifgtel  LastChar,"0",i061
        ifltl   LastChar,"9"+1,i061
                movf    LastChar,W
                call    IdentAddChar
                ifgtel  IdentLength,.4,i063
                        movlw   0x05            ;05 Invalid Char
                        call    errore
                        letl    State,StSeekNewLine
                        goto    nextChar
                ifend   i063
                ifeel   IdentLength,.3,i064
                        movf    Ident+0,W
        ; Conversione da caratteri ASCII "0" "1" "2" "3" "4" "5"
                        addlw   -("0")-(1)
        ; a numeri binari a 8 bit       0xFF  0   1   2   3   4
                        movwf   Ident+0
                        movf    Ident+1,W
                        addlw   -("0")-(1)
                        movwf   Ident+1
                        movf    Ident+2,W
                        addlw   -("0")-(1)
                        movwf   Ident+2

                        ifgtel  Ident+0,4,i068
                                movlw   0x06    ;06 Slot out of range
                                call    errore
                                letl    State,StSeekNewLine
                                goto    nextChar
                        ifend   i068
                        movf    SlotN,W
                        ifgtew  Ident+0,i069
                                movlw   0x08    ;08 No module in slot
                                call    errore
```

```
                        letl    State,StSeekNewLine
                        goto    nextChar
                ifend   i069
                ifgtel  Ident+1,4,i087
                        movlw   0x07    ;07 Chan out of range
                        call    errore
                        letl    State,StSeekNewLine
                        goto    nextChar
                ifend   i087
                ifgtel  Ident+2,4,i089
                        movlw   0x07    ;07 Chan out of range
                        call    errore
                        letl    State,StSeekNewLine
                        goto    nextChar
                ifend   i089

                letl    FSR,Lista
                bcf     STATUS,C
        ;Raddoppia Ident+0 in W.  W vale 0 2 4 6 a seconda di slot number
                rlf     Ident+0,W
                addwf   FSR,F           ;FSR = Lista + 0 2 4 6

                btfsc   Ident+1,1       ;Se riga "3" o "4" (2 o 3 binario 0-based)
                incf    FSR,F           ;FSR = FSR + 1

                clrf    Temp
                        incf    Temp,F          ;Temp = B'0001'
                incf    Ident+2,W
                forw    Looper,i098
                        bcf     STATUS,C
                        rlf     Temp,F
                next    Looper,i098
                bcf     STATUS,C
                rrf     Temp,F
        ; Temp = B'0001' B'0010' B'0100' B'1000' in base a Ident+2
                ifbs    Ident+1,0,i099
                        swapf   Temp,F
        ; Temp = SwapNibbles(Temp) se Ident+1 è riga 2 o riga 4
                ifend   i099

                movf    Temp,W
        ; Setta il bit in Lista corrispondente allo Slot+Channel number
                iorwf   INDF,F          ; riconosciuto nelle righe precedenti
        ifend   i064
        goto    nextChar
ifend   i061

ifeel   LastChar,":",i058
        ; Le liste di canali possono contenere : per intervalli di canali.
        movlw   0x03            ; 03 Unimplemented
        call    errore
        letl    State,StSeekNewLine
        goto    nextChar
ifend   i058

ifeel   LastChar,",",i060
; La virgola "," separa un Slot+Channel number dal successivo
        clrf    IdentLength
; Si azzera Ident in modo da utilizzarla nel riconoscimento successivo
        goto    nextChar
ifend   i060

ifeel   LastChar,")",i062
; Parentesi chiusa: se segue un newline si esegue il comando
        letl    State,StNewLineAfterParClose
        goto    nextChar
ifend   i062

movlw   0x05                    ; 05 Invalid Char
call    errore
letl    State,StSeekNewLine
goto    nextChar
```

In questo stato si può apprezzare un po’ di programmazione in Assembler vera e propria, quando i caratteri numerici che costituiscono i numeri di canale vengono convertiti in numeri

binari senza segno. Su questi si effettuano operazioni di moltiplicazione e addizione per settare o resettare i bit di posto corretto all'interno della Lista di canali. Vengono anche effettuate mascherature con AND e OR logiche, e si fa ricorso all'unica modalità di indirizzamento indiretto disponibile sul PIC, per effettuare correttamente tali operazioni.
Non è questa relazione la sede adatta per approfondire questi aspetti del linguaggio, e si rimanda ad un libro di base sulla programmazione in assembler.

NewLineAfterParClose

Anche dopo aver riconosciuto un comando valido ROUTE:CLOSE seguito da una lista di canali valida, cioè senza errori fino alla parentesi chiusa, non si può ancora eseguire il comando poiché dopo la parentesi chiusa, al posto del carattere new-line, potrebbero apparire ulteriori caratteri non consentiti dalla sintassi. E' necessario perciò uno stato apposito che aspetta il carattere new-line dopo la parentesi chiusa, ed esegue il comando se lo riceve, oppure genera un errore in caso contrario. Questo stato gestisce in maniera efficiente (riutilizzando le parti di codice in comune) i tre comandi ROUTE:CLOSE ROUTE:CLOSE:EXCLUSIVE e ROUTE:OPEN.

```
stNewLineAfterParClose
        ; In questo stato ci si aspetta un NL dopo un ROUTE:CLOSE (@...) o ROUTE:OPEN (@...)
        ; per eseguire finalmente il comando, utilizzando SSParsed Flag(FlExclusive) e Lista
        ; per sapere che tipo di comando è, se il CLOSE è di tipo EXCLUSIVE, e la lista dei
        ; canali da Chiudere/Aprire. Se si riceve un altro carattere che non sia newline,
        ; è un carattere non valido e si genera l'errore di sintassi relativo.
        ifeel   LastChar,0x0A,i045
                bcf     Flag,FlLogic
        ;FlLogic = (SSParsed == SSOpen) || (SSParsed == SSClose)
                ifeel   SSParsed,SSClose,i092
                        bsf     Flag,FlLogic
                ifend   i092
                ifeel   SSParsed,SSOpen,i093
                        bsf     Flag,FlLogic
                ifend   i093
                ifeel   SSDefault,SSRoute,i059
                ifbs    Flag,FlLogic,i059
                        forl    Looper,.8,i065          ; for Looper=0 to 7
                                letl    FSR,Lista-1
                                movf    Looper,W
                                addwf   FSR,F
                                let     Temp,INDF
                                letl    FSR,Relais-1
                                movf    Looper,W
                                addwf   FSR,F

                                ifeel   SSParsed,SSClose,i094
                                        ifbs    Flag,FlExclusive,i081
                        ;Se Exclusive, prima azzera tutti i Relais
                                                clrf    INDF
                                        ifend   i081
                                        movf    Temp,W
                        ; Alza i bit mascherando i vecchi valori con OR
                                        iorwf   INDF,F
                        ; Relais[Looper] = Relais[Looper] OR Lista[Looper]
                                ifend   i094
                                ifeel   SSParsed,SSOpen,i095
                                        comf    Temp,W
                        ; Abbassa i bit mascherando i vecchi valori con AND NOT
                                        andwf   INDF,F
                        ; Relais[Looper] = Relais[Looper] AND NOT( Lista[Looper] )
                                ifend   i095
```

```
                        next    Looper,i065
                        call    aggiornaModuli
                        movlw   0x01    ;01 Completed
                        call    errore
                        letl    State,StStartOfLine
                        goto    nextChar
                ifend   i059
        ifend   i045

        movlw   0x05    ;05 invalid Char
        call    errore
        letl    State,StSeekNewLine
        goto    nextChar
```

Anche questo codice è un buon esempio di programmazione in assembler propriamente detta.

NewLineAfterQM

Similmente alla parentesi chiusa, anche dopo il "?" (Punto interrogativo) di termine query bisogna aspettare new-line per eseguire il comando, e segnalare l'errore di carattere non valido se si riceve altro. Anche l'errore di header non riconosciuto, terminato da un "?" viene riconosciuto in questo stato dal fatto che SSParsed è rimasto al valore iniziale SSUndefined.

Il codice consiste quindi nell'elenco delle istruzioni che implementano l'esecuzione delle differenti query previste per la macchina. Osservare direttamente i commenti nel codice per esaminare le soluzioni pensate per ciascun comando.

```
stNewLineAfterQm
; In questo stato ci si aspetta un newline dopo un "?" qualunque altro carattere è errato
; Ricevuto il newline, si esegue la query. In SSDefault e SSParsed è disponibile il comando
; da eseguire. Se SSParsed è rimasto al valore iniziale SSUndefined, è segno che non
; è stato riconosciuto alcun header valido.
        ifeel   LastChar,0x0A,i018
                ifeel   SSDefault,SSCommon,i025
                ifeel   SSParsed,SSIdn,i025
                        movlw   0x14    ;IDN String     ;*IDN?
                        call    sendString
                        call    newline
                        letl    State,StStartOfLine
                        goto    nextChar
                ifend   i025

                ifeel   SSDefault,SSCommon,i019
                ifeel   SSParsed,SSTst,i019             ;*TST?
                        call    autotest
                        letl    State,StStartOfLine
                        goto    nextChar
                ifend   i019

                ifeel   SSDefault,SSSystem,i039
                ifeel   SSParsed,SSVer,i039             ;SYSTEM:VERSION?
                        movlw   "1"
                        call    trasmit
                        movlw   "9"
                        call    trasmit
                        movlw   "9"
                        call    trasmit
                        movlw   "4"
                        call    trasmit
                        movlw   "."
                        call    trasmit
                        movlw   "0"
                        call    trasmit
                        call    newline
                        letl    State,StStartOfLine
                        goto    nextChar
                ifend   i039
```

```
        ifeel   SSDefault,SSSystem,i077
        ifeel   SSParsed,SSModule,i077          ;SYSTEM:MODULE?
                movf    SlotN,W
                addlw   "0"
                call    trasmit
                call    newline
                letl    State,StStartOfLine
                goto    nextChar
        ifend   i077

        bcf     Flag,FlLogic                    ;ROUTE:CLOSE?  ROUTE:OPEN?
        ;FlLogic = (SSParsed == SSOpen) || (SSParsed == SSClose)
        ifeel   SSParsed,SSClose,i107
                bsf     Flag,FlLogic
                bcf     Flag,FlInvert
        ifend   i107
        ifeel   SSParsed,SSOpen,i108
                bsf     Flag,FlLogic
                bsf     Flag,FlInvert
        ;Gli 1 e 0 inviati da sendINDF verranno complementati
        ifend   i108
        ifeel   SSDefault,SSRoute,i082
        ifbs    Flag,FlLogic,i082
        ;ROUTE:CLOSE? oppure ROUTE:OPEN?
                ifeel   SlotN,.0,i083
                        movlw   "0"
                        call    trasmit
                        movlw   "0"
                        call    trasmit
                        call    newline
                        letl    State,StStartOfLine
                        goto    nextChar
                ifend   i083
; Anziché implementare una routine di moltiplicazione di SlotN per 16, e successiva
; conversione da binario a decimale, si è preferito un mero elenco di if-then
; per ogni possibile valore di SlotN poiché è questo l'unico caso di utilizzo
; della conversione binario-decimale. Se ci fossero più casi, si risparmierebbe
; più memoria con una routine dedicata. Essendo pochi i casi, si risparmia memoria con
; l’approccio ad elenco di if-then
                ifeel   SlotN,.1,i084
                        movlw   "1"
                        call    trasmit
                        movlw   "6"
                        call    trasmit
                ifend   i084
                ifeel   SlotN,.2,i085
                        movlw   "3"
                        call    trasmit
                        movlw   "2"
                        call    trasmit
                ifend   i085
                ifeel   SlotN,.3,i086
                        movlw   "4"
                        call    trasmit
                        movlw   "8"
                        call    trasmit
                ifend   i086
                ifeel   SlotN,.4,i090
                        movlw   "6"
                        call    trasmit
                        movlw   "4"
                        call    trasmit
                ifend   i090

                clrf    Looper
        ;For Looper = 1 to SlotN
                repeat  i091
        ;Non si è potuta usare la macro for, perché conta a scendere
                incf    Looper,F        ;invece a noi interessava contare a salire
                        letl    FSR,Relais-2            ;FSR = Relais(2*(Looper-1))
                        bcf     STATUS,C
                        rlf     Looper,W
                        addwf   FSR,F
                        call    sendINDF
                ; Invia 8 0 o 1 in base ai bit di INDF
                        incf    FSR,F
                        call    sendINDF
                until2  ifee,Looper,SlotN,i091   ;Fine del ciclo For su Looper
```

```
                call    newline
                letl    State,StStartOfLine
                goto    nextChar
        ifend   i082

        movlw   0x04    ;04 Undefined header
        call    errore
        letl    State,StStartOfLine
        goto    nextChar
ifend   i018

movlw   0x05    ;05 invalid Char
call    errore
letl    State,StSeekNewLine
goto    nextChar
```

Notare che la stringa di risposta a SYSTem:VERsion? è memorizzata carattere per carattere direttamente nel codice di esecuzione: essendo una stringa corta, non si sarebbe risparmiata memoria memorizzandola nella look-up table. La risposta ad *IDN?, invece, essendo lunga, è più efficientemente memorizzata nella look-up table.

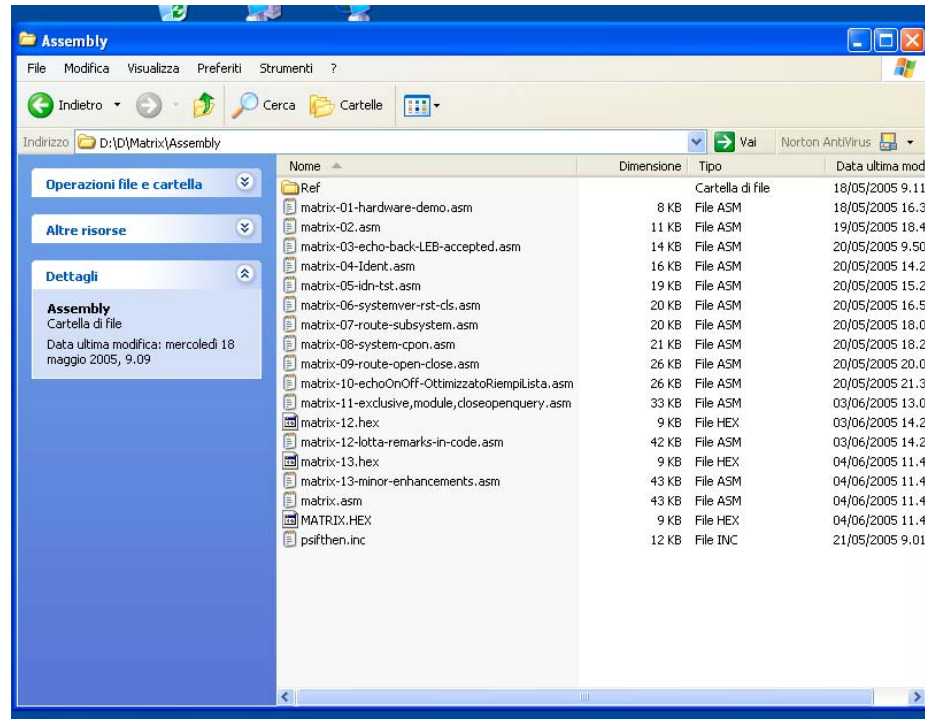
Approfondimenti sul parser

La descrizione del parser data è sufficientemente dettagliata per poterne comprendere ogni intimo dettaglio implementativo, ma presuppone che il lettore già conosca le modalità di implementazione di un tipico parser ad una sola passata per sintassi con pochi costrutti. Per approfondimenti su questo tipo di parser, si rimanda al mio lavoro, già citato in una nota a pié di pagina, “Generazione dell’albero corrispondente ad un documento XML mediante un parser scritto in Java”, che utilizza tecniche simili e contiene descrizioni molto più approfondite delle tematiche riguardanti i parser.

Revisioni e numero di versione

Ogni volta che si realizza un programma, è una buona idea salvare separatamente ogni versione funzionante, prima di apportare modifiche che potrebbero far perdere di funzionalità il programma. Ciò è particolarmente vero per modifiche e ristrutturazioni più radicali, e molto importante in casi in cui il debug è molto difficoltoso, come per un microcontrollore.

Salvare ogni versione funzionante è però inutile e rende la cartella di lavoro ingestibile. L’esperienza insegna a non eccedere con troppi o troppo pochi cambi di numero di versione. Una buona regola pratica è osservare il tempo dedicato alla stesura del codice, che è un indice della complessità reale, con la propria esperienza, di implementazione, più che la complessità teorica o presunta. L’obiettivo è non perdere e non dover rifare il proprio lavoro, e avere sempre a portata di mano un file compilato per il quale si è certi del funzionamento, che non

contenga “variazioni per debug”[61], e che possa far funzionare immediatamente il circuito per una dimostrazione o per controllare che l’hardware non abbia problemi, e che se le cose non stanno funzionando come sperato è colpa del firmware.

La cartella con versioni successive, numerate, del file sorgente,
nominate con le caratteristiche via via aggiunte,
e qualche file .hex compilato dal funzionamento certo

Conviene conservare tutte le versioni del file sorgente, ed è opportuno numerarle fin dall’inizio con semplici cardinali, anziché utilizzare nomi poco rappresentativi come “nuova” “nuovissima” “funzionante” “definitiva” “super-definitiva” “super-tantissimo-definitiva-finale”. E’ opportuno anche conservare qualche versione di file compilati per una riprogrammazione rapida del dispositivo senza dover riaprire l’ambiente di sviluppo e rieseguire la compilazione. In caso di malfunzionamento della macchina, se si ha il dubbio se sia l’hardware o il software il problema, si riprogramma al volo il chip senza il dubbio che si sia ricompilato il codice con una configurazione differente del compilatore.

[61] Quelle piccole variazioni al flusso normale di codice volte a testare direttamente il punto che crea problemi: è possibile a volte dimenticare di rimettere a posto le cose, e può essere difficile cercare la piccola modifica in tutto il corpo del programma, se ci si è dimenticati di averla fatta. In questi casi può essere più produttivo riprendere una versione vecchia e funzionante e ricostruire l’ultima implementazione da zero, ora che ci si è chiarito meglio

Revisioni del codice

Un buon sistema di numerazione delle versioni è anche utile in caso di revisioni del codice. Il processo di revisione, miglioramento e debug di un software non ha mai termine. Nel nostro caso, dopo l'utilizzo pratico del circuito col Terminale di Windows, con LabView e Matlab, si sono ideate nuove funzionalità utili e modificata leggermente l'implementazione di alcune.

Queste revisioni possono essere spesso pesanti e difficili, anche se il codice è molto ordinato. Modificare un pezzo di programma può produrre errore inaspettati in altre parti del programma, che si basavano sulla presenza (o assenza) di operazioni in altre zone.

Per questo è importantissimo progettare fin dall'inizio tutte le funzionalità di cui si ritiene di aver bisogno: si dedica più tempo e lavoro all'inizio, ma se ne evita dopo. In caso di revisioni profonde, è opportuno progettare adeguatamente anche l'approccio migliore da seguire per introdurre le variazioni nel codice già pronto, in modo che l'operazione di codifica e debug sia il più possibile diretta e rapida. Progettare prima di codificare anche le revisioni, dunque.

Revisioni "postume"

Segue qualche esempio di revisione alle modalità di funzionamento la cui necessità si è evidenziata solo dopo l'uso con altri programmi, a testimonianza che spesso in un progetto si è costretti a tornare indietro e rivedere un aspetto che si riteneva definitivo e riaprire un discorso che si riteneva chiuso.

Inizialmente il circuito, all'accensione, dopo il messaggio *02 Hallo*, ne inviava un altro con il numero di moduli trovati, del tipo *X module(s) found*. Se questo era piacevole da osservare sul Terminale di Windows, era fastidioso per l'utente che aspirava ad utilizzare la macchina in un contesto automatico, poiché lo costringeva a leggere una ulteriore stringa, dopo il messaggio Hallo, per di più di sintassi differente dalle altre. Si è preferito invece, per facilitare il lavoro all'utente/programmatore, che la macchina rispondesse sempre con un'unica stringa ad ogni comando ricevuto. L'informazione sul numero di moduli presenti era comunque utile, e si è deciso di estendere il set di comandi con un comando proprietario: SYSTEem:MODule? che restituisce appunto il numero di moduli connessi. Il comando è stato inserito su un SubSystem già presente.

Il beep del messaggio di errore durava inizialmente 300 ms, per essere udibile con facilità, e il messaggio di errore veniva inviato DOPO il beep. Dopo l'uso della macchina in LabView, in cui era più facile utilizzare il tempo di attesa massimo ogni volta, anziché il minimo con

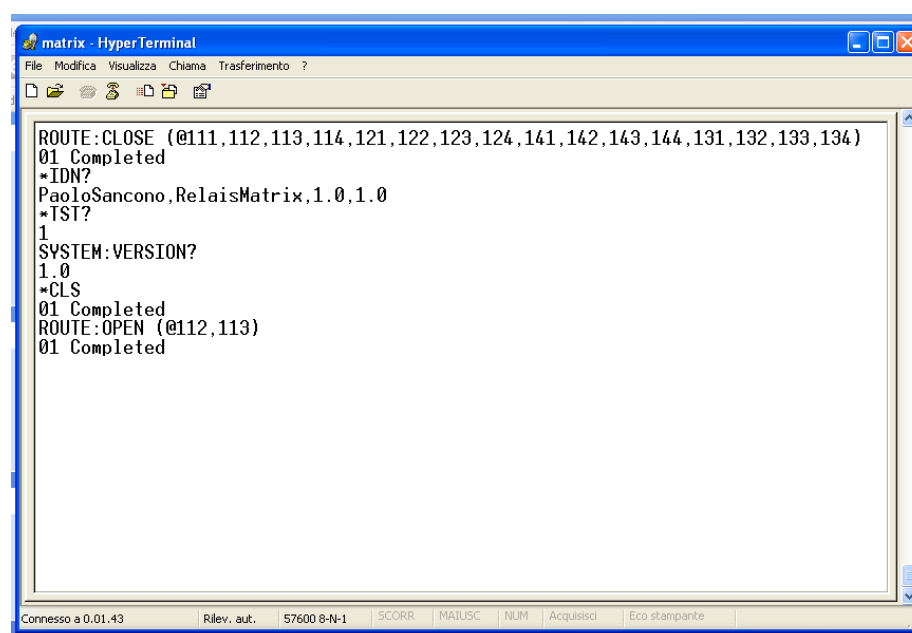
---

come procedere, piuttosto che cercare di rimettere a posto un codice che è stato modificato in più punti fino a perdere l'orientamento.

qualche complicazione al codice illustrata nell'appendice "Manuale dell'utente", si è deciso di accorciare il beep a 100ms soltanto, e il messaggio di errore lo si invia PRIMA del beep, così da consentire al PC di fare altro, anziché restare in attesa, mentre il MCU si dedica alla gestione del buzzer.

Gli errori hardware venivano comunicati all'accensione dopo il messaggio *02 Hallo*. Ora lo sostituiscono, in modo da mantener fede all'impegno di inviare sempre e solo un'unica stringa di risposta ai comandi[62].

---

[62] Ogni regola ha le sue eccezioni. C'è un unico caso in cui l'utente del PC deve leggere una stringa ulteriore dopo una risposta: se l'autotest fallisce, e *TST? restituisce "+0", in un messaggio successivo si comunica l'errore hardware occorso, in modo da poter distinguere tra le varie cause. Quindi il programma del PC deve effettuare una nuova lettura se riceve "+0" in risposta a "TST?". Se la cosa non è gradita si può modificare facilmente il firmware eliminando la comunicazione dell'errore occorso e lasciando solo la generica risposta "+0": autotest fallito.

Capitolo 5 - Utilizzo

Terminale di Windows

Il terminale di Windows, presente in ogni versione del sistema operativo Microsoft, consente di comunicare facilmente con una porta seriale del PC. E’ sufficiente configurare i parametri di comunicazione, per ottenere la visualizzazione a schermo dei caratteri ricevuti e l’invio sulla porta di comunicazione dei caratteri digitati su tastiera, in emulazione terminale.

Esistono però tre problemi che rendono difficoltosa l’interazione:

- I caratteri digitati su tastiera vengono inviati alla porta, ma non vengono stampati a schermo;
- Ricevendo il carattere new-line, il cursore finisce alla riga inferiore, ma senza tornare a capo, poiché in Windows l’andata a capo è rappresentata dalla coppia di caratteri CR/LF;
- Premendo INVIO, viene inviata la coppia CR/LF anziché solo LF, e inizialmente il circuito accettava come terminatore di riga solo LF, e avrebbe riportato CR come carattere non valido.

```
ROUTE:CLOSE (@111,112,113,114,121,122,123,124,141,142,143,144,131,132,133,134)
01 Completed
*IDN?
PaoloSancono,RelaisMatrix,1.0,1.0
*TST?
1
SYSTEM:VERSION?
1.0
*CLS
01 Completed
ROUTE:OPEN (@112,113)
01 Completed
```

Una sessione di lavoro con Relais Matrix al terminale di Windows

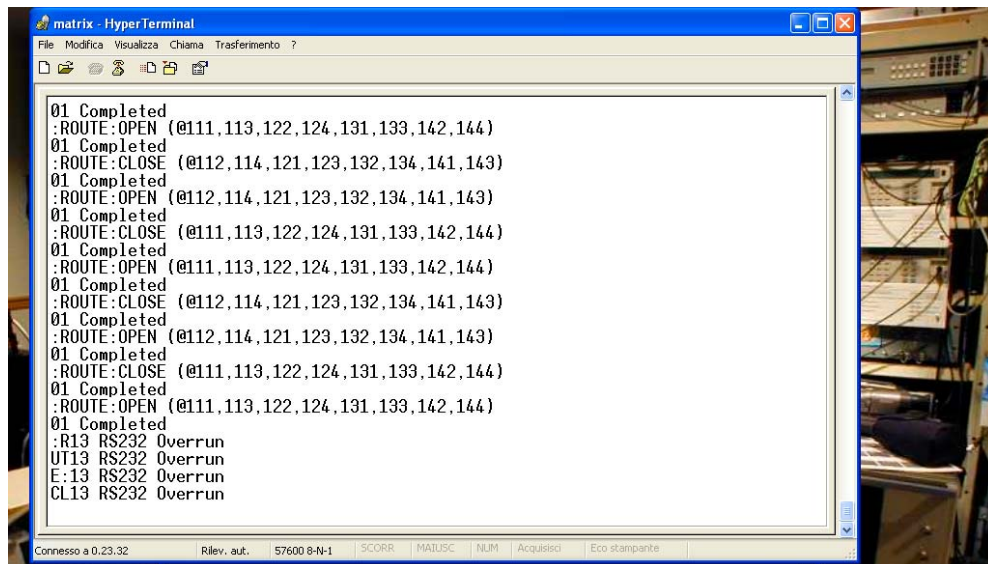
Per ovviare a questi inconvenienti, si è aggiunta al ciclo principale del firmware la possibilità di scartare i CR ricevuti, in modo da accettare come terminatore di riga anche la coppia CR/LF, e la gestione di due combinazioni di tasti fuori standard:

- CTRL-E: Attiva/Disattiva l’echo dei caratteri ricevuti;

- CTRL-F: Attiva/Disattiva l’emissione di CR/LF anziché LF da solo.

Queste due combinazioni di tasti sono state introdotte per facilitare il debug del firmware lavorando al terminale di Windows, per questo non se ne è curata l’implementazione “elegante” (per esempio con altri comandi riservati), a favore di una implementazione “fault-proof”, ovvero certamente funzionante, poiché lo scopo principale era favorire il debug, non estendere il numero di comandi accettati.

L’echo dei caratteri permette al contempo di leggerli sullo schermo del terminale di Windows, e di verificare che la trasmissione seriale stia funzionando, e il firmware stia eseguendo il ciclo principale anziché essersi smarrito chissà dove.

Con copia/incolla di lunghe sequenze di comandi, si è potuto verificare che l’approccio scelto per il controllo di flusso funziona bene, anche senza nessuna cura particolare da parte del programmatore al PC.

Solo sporadicamente si è riusciti a generare l’errore di overrun della seriale

Evidentemente anche il terminale di Windows inserisce brevi pause di trasmissione mentre riceve caratteri, con lo scopo di visualizzare al posto corretto le risposte, inframmezzate con le richieste, per far fronte a dispositivi half-duplex e non impiastricciare i caratteri sullo schermo, caratteristica essenziale per un programma di emulazione terminale.

Matlab

Se il terminale di Windows è lo strumento ideale per il debug del firmware, fornendo un accesso quanto più diretto possibile alla porta seriale, non è però la scelta migliore se si desiderano funzioni di scripting avanzato.

In tal caso è molto utile avere delle librerie di funzioni di accesso a Relais Matrix all'interno di un ambiente di calcolo scientifico come il Matlab[63], che è in grado di interfacciarsi con altri strumenti ATE.

Funzioni di basso livello per Matlab

Sono stati scritti due ordini di funzioni, memorizzate in m-files (il tipo di file standard per script e funzioni Matlab): il primo di più basso livello, che consente di inviare e ricevere comandi in forma testuale alla macchina, tramite la porta seriale.

- **MatrixInit**: Crea una variabile globale, MatrixSp, che contiene l'handle per un oggetto di accesso alla porta seriale utilizzato poi da tutte le altre funzioni disponibili. MatrixInit va richiamato in interattiva all'inizio della sessione di lavoro.
- **MatrixDone**: Rilascia le risorse di memoria occupate dall'oggetto MatrixSp. MatrixDone va richiamato in interattiva alla fine della sessione di lavoro.
- **MatrixWrite**: Accetta come unico parametro una stringa, che verrà inviata alla macchina. E' la funzione stessa che si incarica di aggiungere il carattere di Newline in coda alla stringa.
- **MatrixRead**: Legge le stringhe ricevute dalla macchina, seguendo una politica FIFO. La funzione stessa si incarica di estrarre i terminatori di riga e fornire una stringa per volta.
- **MatrixFlush**: Svuota l'intero buffer di lettura FIFO, in modo da ripartire con un buffer certamente vuoto e poter leggere le risposte in ordine.

Funzioni di medio livello per Matlab

Funzioni di medio livello facilitano l'utilizzo del circuito per gli scopi più tipici (apertura e chiusura di relais).

- **MatrixClose:** Accetta come unico argomento un vettore di numeri interi. Chiude i relais corrispondenti sul primo modulo della macchina. I numeri sono a due cifre, la prima rappresenta la riga, la seconda la colonna.
- **MatrixOpen:** Apre i relais selezionati.
- **MatrixCloseExclusive:** Chiude i relais selezionati, dopo averli aperti tutti.

Le funzioni in m-files di basso livello sono state piuttosto semplici da implementare, facendo ricorso agli oggetti di gestione della porta seriale già disponibili in Matlab. Anche le funzioni

[63] Il Matlab è stato scelto poiché è stato l'ambiente di calcolo utilizzato durante il corso di "Misure Elettroniche".

di medio livello sono di implementazione facile, poiché si limitano a creare una stringa di comando opportuna, da inviare poi con le funzioni di basso livello.
Osservando il codice sorgente delle funzioni in Matlab, i cui listati sono riportati in appendice, è immediato tradurre tali funzioni in un qualsiasi altro linguaggio di programmazione che consenta l'utilizzo di una porta seriale presente sul PC.

Un esempio di utilizzo: pizzica.m

Just for the fun of it[64] è stato scritto uno script dimostrativo che invia a ciclo continuo una sequenza di aperture e chiusure relais alla macchina.
Per la temporizzazione ci si è volutamente affidati esclusivamente alla funzione pause del Matlab. Questo script è volto a dimostrare l'inaffidabilità di tale approccio che non consente temporizzazioni precise, anche perché il sistema operativo sottostante, Windows XP, non è di tipo real-time, ma multi-tasking, e può togliere il controllo al programma utente in qualunque momento.
Con lo script, che prevede accensioni e spegnimenti contemporanei di un numero continuamente variabile di relais, fino ad 8, si dimostra anche che il circuito è stato realizzato correttamente ed è insensibile ai disturbi elettromagnetici e ai problemi derivanti da carichi induttivi considerevoli durante aperture e chiusure continue.
Tale script è anche indicato per il "burn-in test", ovvero un test stressante per l'hardware in cui si verifica che tutto funzioni a dovere con le sollecitazioni massime possibili, per un certo tempo. Conviene non esagerare poiché i relais si scaldano, diventano appena più fragili, e finiscono per rompersi con molti azionamenti a raffica.

[64] Trad. letterale: Per il puro divertimento nel farlo.

Parentesi mistica

Il nome dato allo script “pizzica”, è quello di una danza popolare del Salento, la regione a Sud delle Puglie, mia terra natale. Lo script tenta di imitare il ritmo ossessivo di tale danza, suonata tradizionalmente con tamburelli a cornice.

Tamburelli a cornice, tradizionalmente impiegati nel Salento per accompagnare la “Pizzica”, una danza popolare arcaica di queste terre, ancora in voga tra i giovani.

L’orecchio attento di un suonatore di tamburello[65], riconosce gli errori di *jitter* introdotti dal sistema operativo e dal Matlab, come fossero errori di un suonatore alle prime armi che va continuamente fuori tempo.

Per un amante della storia dell’informatica, quale io sono, sentire il “Klic-Klac” dei relais azionati automaticamente da una macchina che memorizza non più di 2k di parole di codice, riporta alla memoria il Tech Model Railroad Club, al palazzo 26 del Massachussetts Institute of Technologies, nell’inverno del 1958-1959, quando sotto l’immenso plastico ferroviario un grosso ammasso di commutatori rotativi e relais della compagnia telefonica locale[66], col suo “Chunka-chunka”[67] commutava automaticamente gli scambi, portando i modellini di treni a

[65] Per ascoltare un suonatore di tamburello mentre esegue la Pizzica, si consiglia il CD-Demo gratuito dei MoTaCuntu, gruppo di musica popolare con il quale ho suonato tamburelli a cornice, duff, tammorre e flauto per diversi anni.

[66] Un Computer IBM PDP-2 costava ancora troppo, persino per il MIT, che l’acquistò l’anno dopo, e ci si accontentava di commutatori rotativi e circuiti a relais, forniti gratuitamente all’università dalla compagnia telefonica locale.

[67] Parola onomatopeica inventata direttamente dai realizzatori dell’impianto.

destinazione, senza farli scontrare tra loro e cercando i percorsi migliori, dopo aver impostato la destinazione con il disco combinatore di un telefono collegato all'impianto[68].

Il *trolley terminal* in scala e una *open house* al TMRC

Qualunque ne sia la fonte, far funzionare un circuito in maniera automatica suscita in ogni caso forti emozioni, tanto nell'autore che nel pubblico presente ad una dimostrazione, come se anche la messa in opera di un circuito funzionante sia una forma d'arte possibile.

LabView

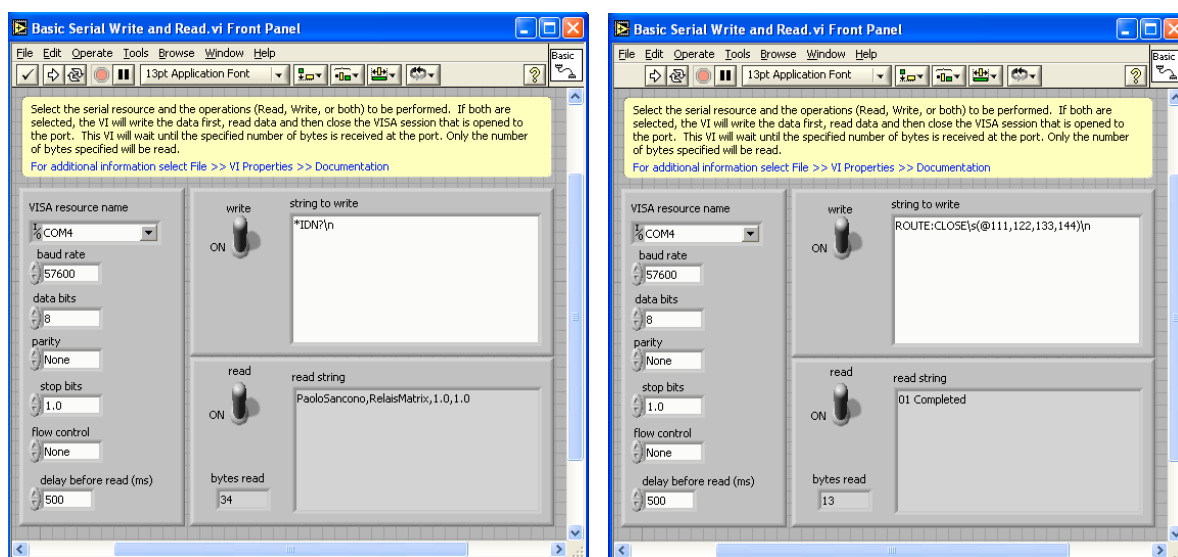
Mentre il Matlab è l'ambiente ideale per utilizzare proficuamente il circuito in un contesto di complesse misure automatiche, si può avere a volte la semplice necessità di dover comandare interattivamente il circuito con un pannello di comandi virtuale, visto che il circuito stesso non è dotato di un pannello reale composto da interruttori, manopole e quant'altro.

Il LabView, ambiente di sviluppo di programmi grafico, prodotto dalla National Instrument, è lo stato dell'arte per la realizzazione di pannelli di controllo virtuali. Un programma in LabView viene infatti chiamato Virtual Instrument, proprio per il grande impatto visivo e funzionalità avanzate della sua interfaccia utente, che rispecchia fedelmente quella di un comune strumento da laboratorio.

[68] Per maggiori informazioni sul Tech Model Railroad Club e la nascita della programmazione come la conosciamo oggi, si veda il libro di Steven Levy, "Hackers – Gli eroi della rivoluzione informatica", Shake ed. Underground.

Basic Serial Write and Read

La versione di LabView utilizzata è stata la 7.0 Student Edition. Per prendere confidenza con l'uso della porta seriale, è stato utilizzato un VI di esempio già pronto: "Basic Serial Write and Read", che sembrava fatto apposta per l'uso con questo circuito.

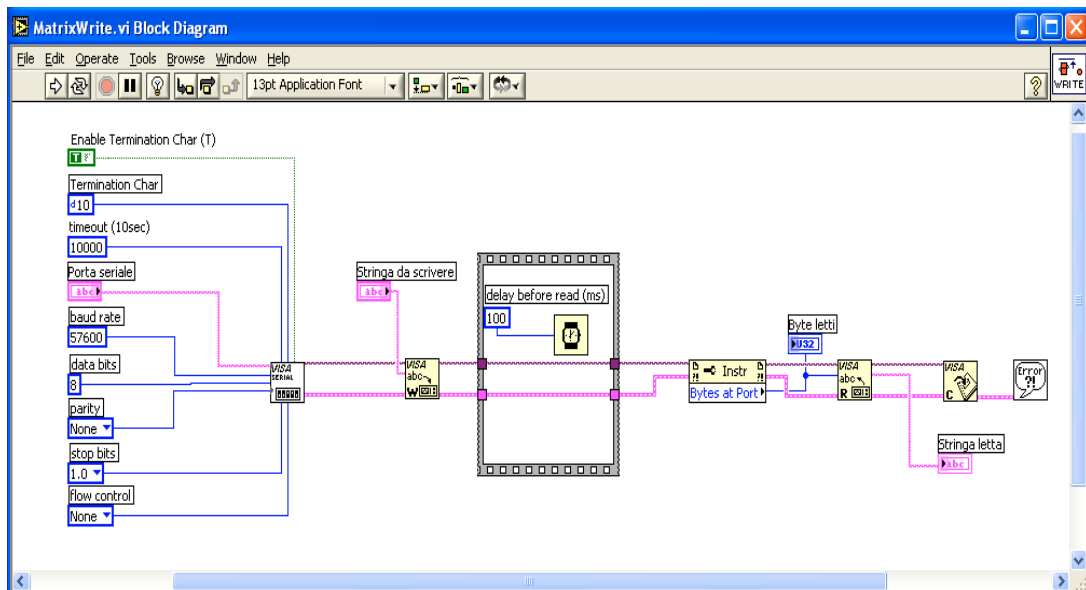
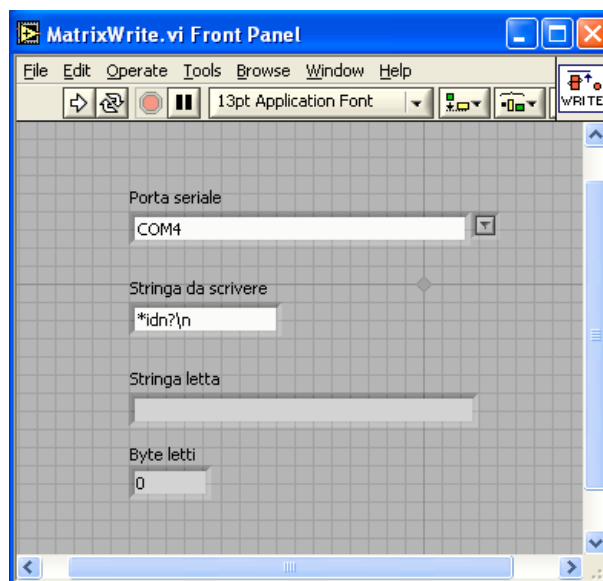
Il VI è in grado di inviare una stringa sulla porta seriale, attendere un certo lasso di tempo, e rileggere una stringa dalla seriale, tipicamente la risposta di uno strumento da laboratorio. LabView è in grado di estrarre righe successive separate dal carattere new-line nella risposta.

VI dimostrativo, già fornito con Labview, sull'uso della porta seriale con strumentazione.

Il VI di esempio ha funzionato senza modifiche con il mio strumento, come evidenziato nelle esecuzioni di prova qui sopra: nell'esempio a sinistra è stata richiesta la stringa identificativa, nell'esempio a destra sono stati chiusi i relais della diagonale principale sul primo modulo, a conferma che il circuito rispetta gli standard di comunicazione industriali.

MatrixWrite

Con lievi modifiche, il VI di esempio si prestava a diventare il SubVi di comunicazione con la macchina da utilizzare in VI più avanzati. Ho chiamato il SubVi di comunicazione MatrixWrite, e ho disegnato una icona rappresentativa raffigurante un relais e un Led.

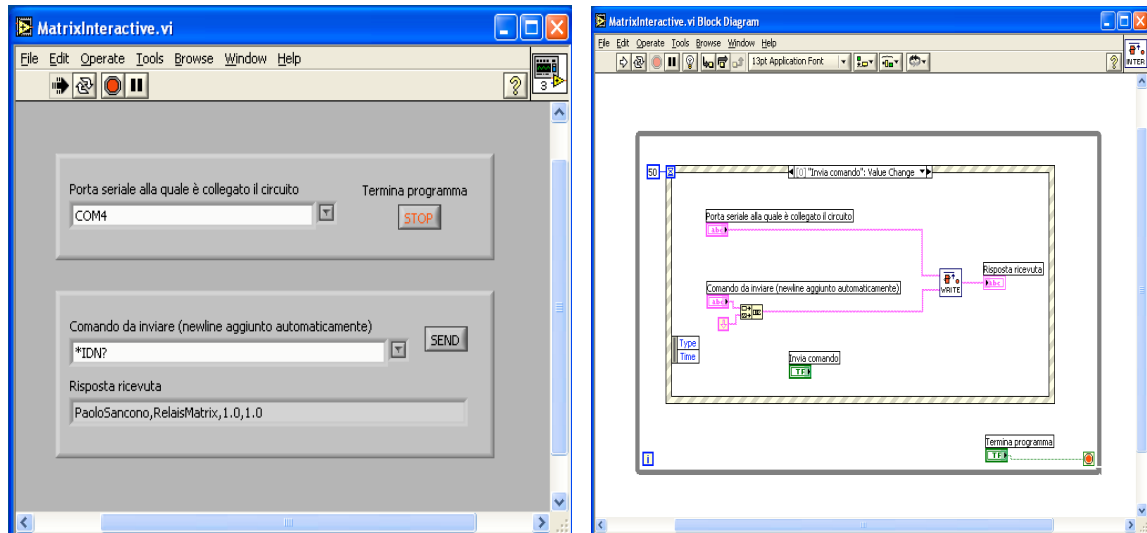
In LabView, infatti, i sottoprogrammi vengono rappresentati con icone nei programmi di livello superiore che li utilizzano, ed è cura del programmatore disegnare tali icone. MatrixWrite gestisce autonomamente gli errori della porta seriale, mostrando una finestra di dialogo, e quindi non fornisce nessun error-out in uscita, e non richiede la gestione degli errori

nei livelli superiori. MatrixWrite richiede in ingresso la stringa da inviare allo strumento, e la porta di comunicazione alla quale il circuito è collegato. In uscita restituisce la stringa di risposta letta dallo strumento.

Block diagram di MatrixWrite.VI

Font Panel di MatrixWrite.VI

Come si osserva in figura, i programmi in LabView vengono disegnati sullo schermo con il mouse, anziché essere immessi da tastiera. Questo VI non viene utilizzato così come è, ma come SubVi da utilizzare altrove. Il suo front panel, quindi, non verrà mai visualizzato a schermo né tanto meno utilizzato, se non per operazioni di debug.

MatrixWrite elimina il line-feed in coda alla risposta ricevuta, ma non aggiunge automaticamente il line-feed nella stringa da inviare allo strumento.

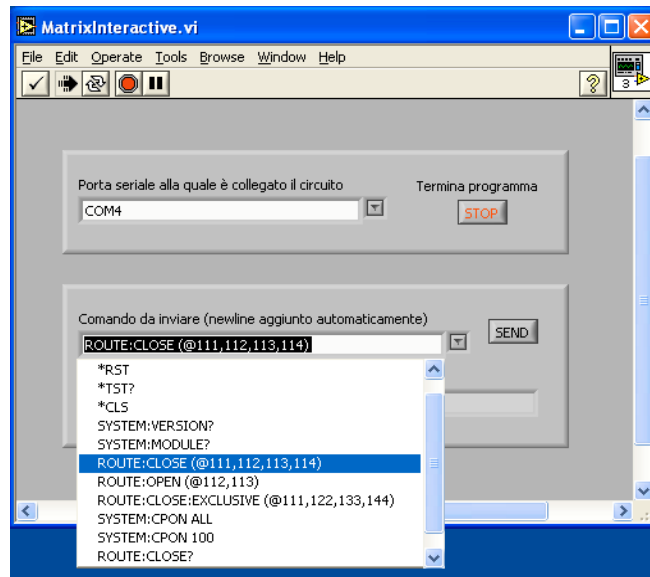
MatrixInteractive

Il passo successivo è stato scrivere un VI che consentisse di immettere comandi generici e leggere le risposte fornite dallo strumento.

Front Panel e Block Diagram di MatrixInteractive.VI

Il programma è di uso immediato: si seleziona la porta di comunicazione desiderata, e si lancia il programma. Una volta scritto il comando da eseguire, clickare semplicemente su Send ed osservare gli effetti prodotti e la risposta data dal circuito. Clickare su termina programma per uscire.

Ciclo while principale e condizione di uscita

Il programma è costituito da un ciclo While in esecuzione continua, che termina clickando sull'apposito bottone. In figura il particolare del codice che implementa questa soluzione, la più tipica dei programmi LabView.

Si osservi poi un altro costrutto LabView: la cornice con il gestore degli eventi. Il codice all'interno di tale cornice viene eseguito quando si genera l'evento indicato in testa alla cornice: cioè la pressione del pulsante "Invia comando", che ha etichetta "Send" sul Front Panel.

La drop-down box con esempi di comandi disponibili

L'utilizzo del VI è facilitato dalla presenza di una drop-down box con esempi di comandi validi. Un effetto collaterale, molto gradito, di questa drop-down box, è il completamento automatico del comando in immissione. Per una prova interattiva, provare ad eseguire in sequenza:

- ROUTE:CLOSE (@111,112,113,114)

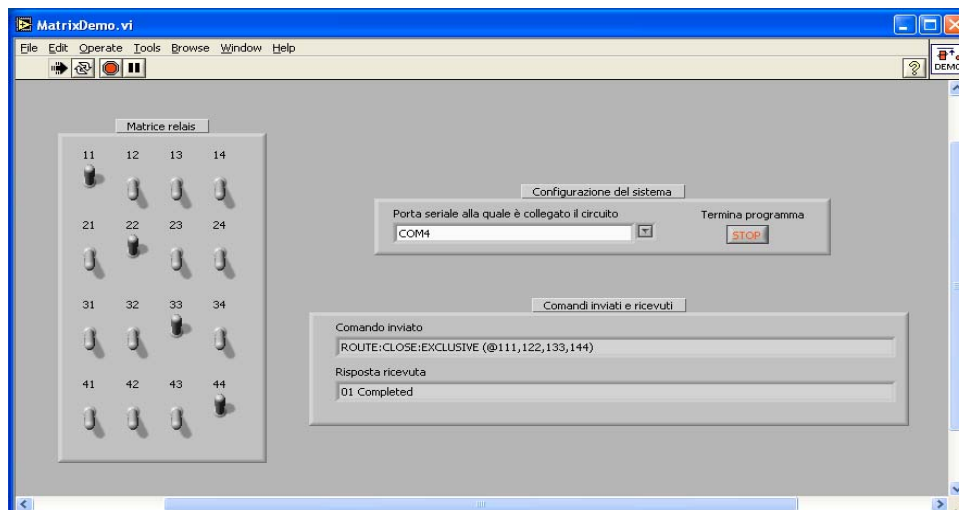
  Verranno chiusi i relais della prima riga sul primo modulo.
- ROUTE:OPEN (@112,113)

  Verranno aperti il secondo e terzo relais della prima riga, e resteranno chiusi il primo e l'ultimo.
- ROUTE:CLOSE:EXCLUSIVE (@111,122,133,144)

  Verranno chiusi i relais della diagonale principale, aprendo tutti gli altri.
- SYSTEM:CPON ALL

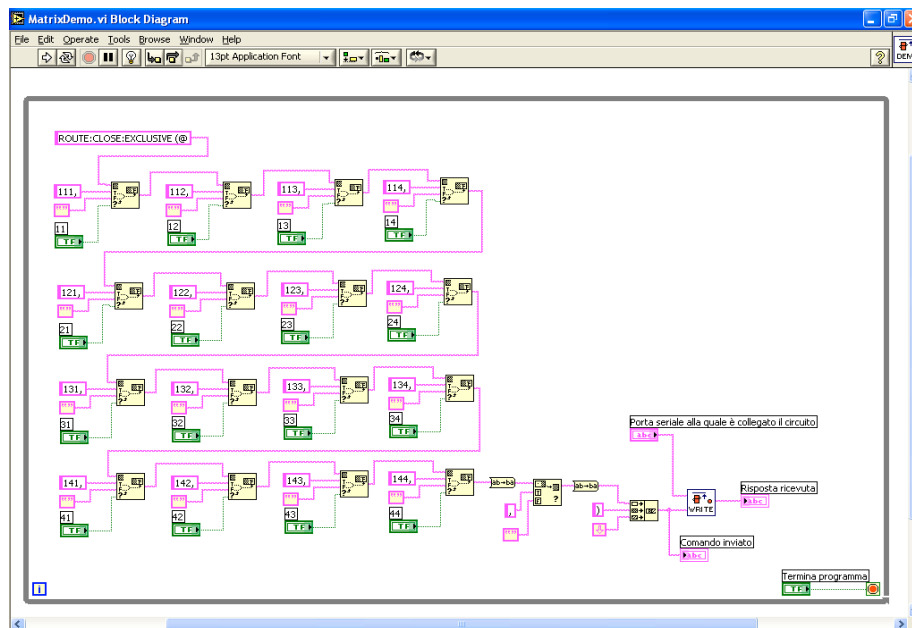
  Tutto si riporterà alle condizioni iniziali, vale a dire tutti i relais aperti

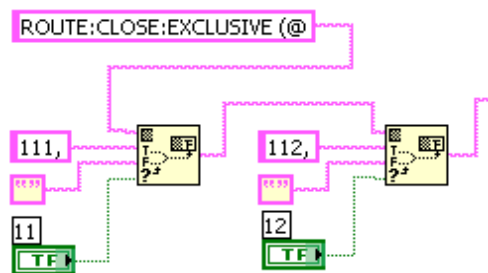
MatrixDemo

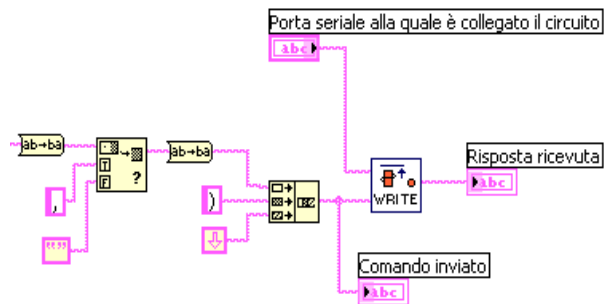
MatrixInteractive.VI è comodo e semplice da usare, ma richiede ancora l'immissione di comandi da tastiera. Si può rendere l'interazione con la macchina ancora più semplice con un VI come MatrixDemo.VI.

Front Panel di MatrixDemo.VI

Il pannello frontale è realmente intuitivo e facile da usare[69]: presenta 16 interruttori virtuali, azionando i quali si modifica lo stato dei relativi relais sul circuito. Il VI mostra pure, per scopi didattici, i comandi generati e le risposte fornite dalla macchina.

Il programma, realizzato senza troppi sforzi in LabView, dà realmente una impressione di alta professionalità, grazie alla potenza dell'ambiente di sviluppo della National Instruments, che ha come suo punto di forza, assieme all'interazione con la strumentazione da laboratorio, e una nutrita serie di funzioni già pronte per il trattamento di dati acquisiti, anche a fini di controllo, proprio le accattivanti GUI dei VI generati.

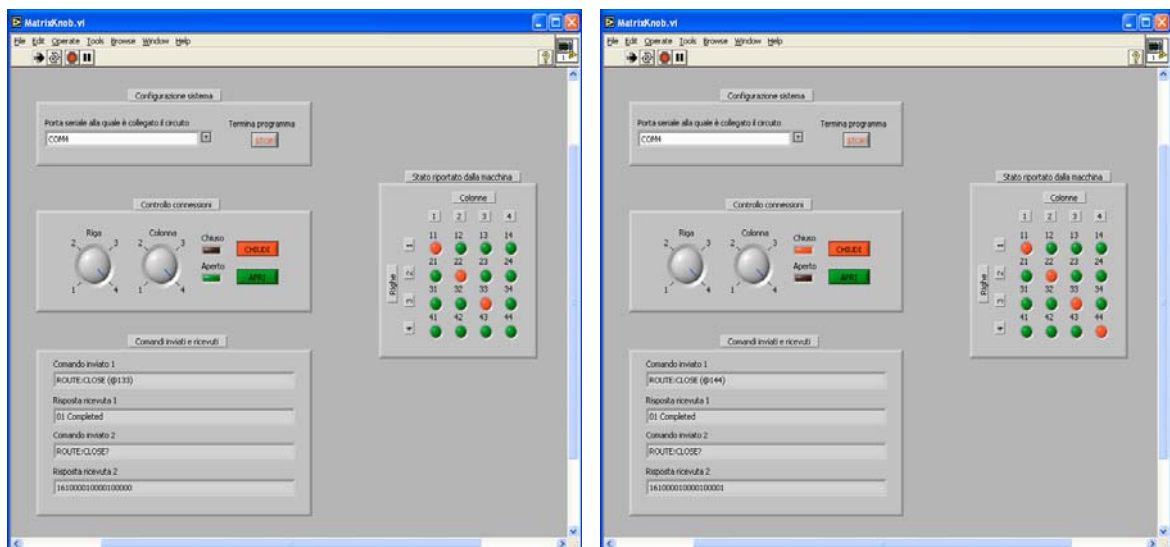
[69] In inglese diremmo, con una dizione più diretta e immediata, che in italiano risulterebbe però lievemente offensiva, "a prova di idiota".

Block Diagram di MatrixDemo.VI

Il Block Diagram è costituito dalla generazione della stringa di comando opportuna, realizzata a partire dallo stato degli interruttori, e dall'invio di questa allo strumento tramite il SubVi MatrixWrite.

Dettaglio della generazione della stringa.

Lo stato di ogni interruttore produce l'accodamento del numero di canale corrispondente, seguito da virgola, alla stringa in formazione, oppure una stringa vuota (non accoda niente).

Conclusione della stringa e invio allo strumento

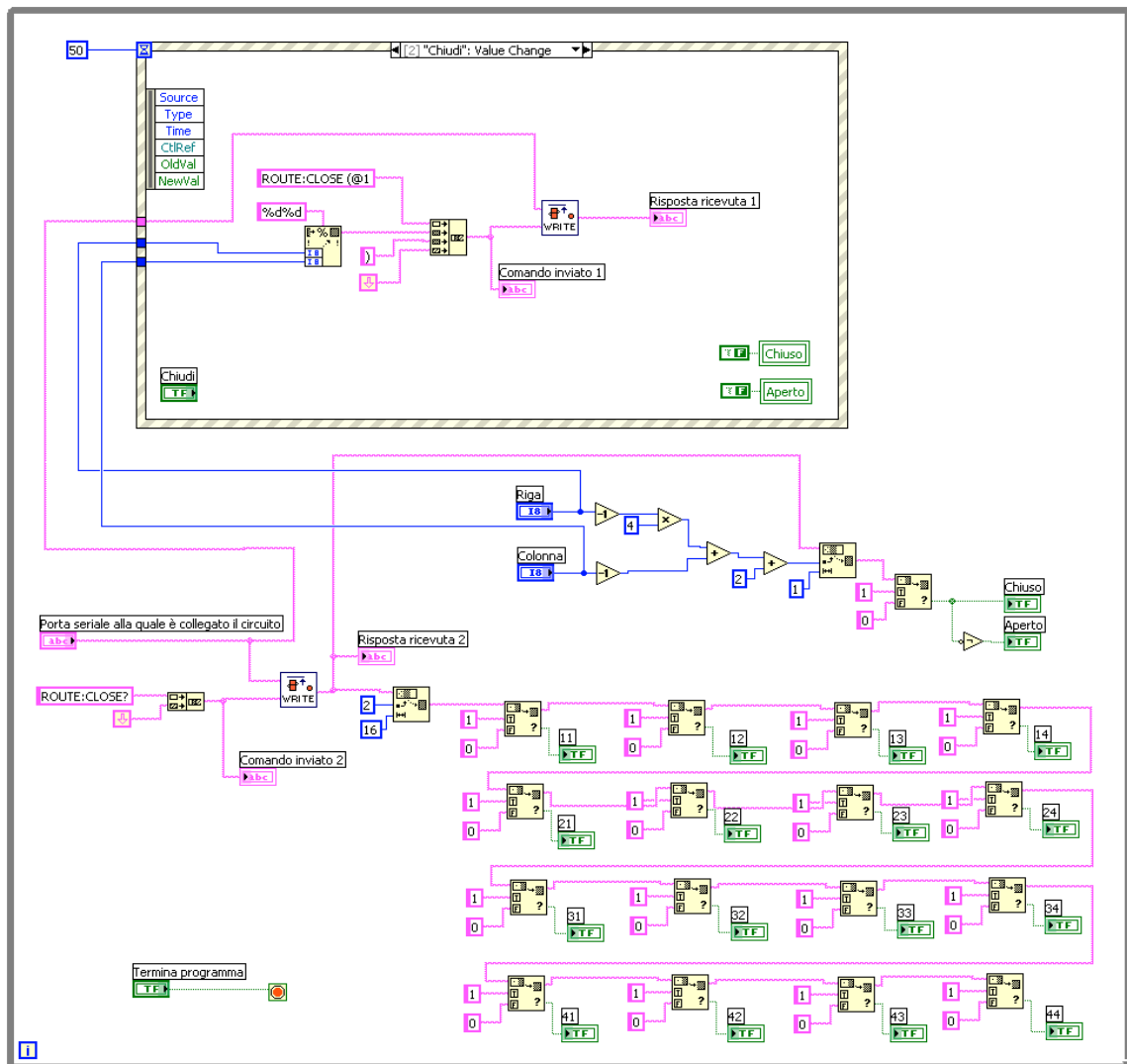
Al termine viene eliminata l’ultima virgola dalla stringa, aggiunta una parentesi chiusa e il carattere di new-line, e la stringa generata viene inviata allo strumento mediante il SubVi MatrixWrite.

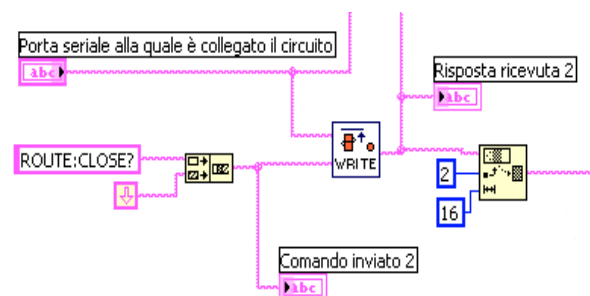
MatrixKnob

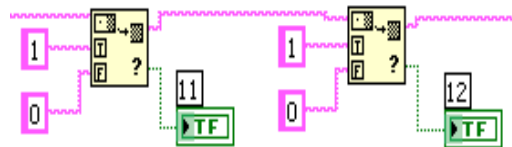
Se è vero che MatrixDemo.VI funziona bene, è anche vero che non effettua nessun controllo di comando eseguito correttamente. Inoltre ha il difetto che, appena avviato, impone lo stato degli interruttori al circuito, anziché leggere lo stato attuale del circuito e partire da quello con le modifiche. Inoltre se è piacevole la corrispondenza 1-a-1 con gli interruttori presenti a schermo rispetto ai relais presenti sul circuito, è anche vero che tale rappresentazione è fuorviante: ha più senso evidenziare le connessioni create tra ingressi e uscite (righe e colonne) tramite i relais che non lo stato di apertura e chiusura dei relais in quanto tali.

Front Panel di MatrixKnob subito prima e subito dopo la chiusura dell’ultimo relais della diagonale principale.

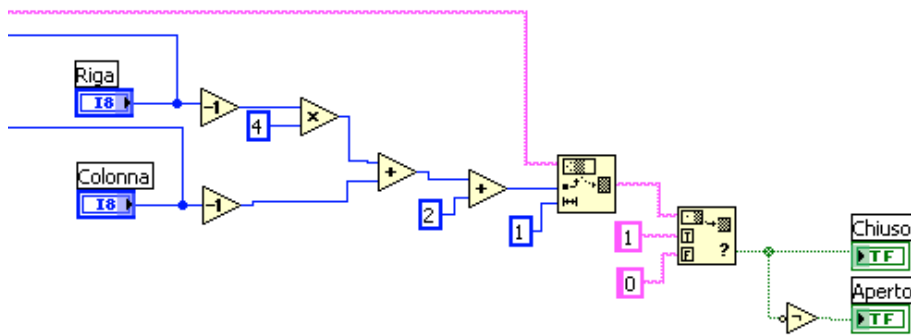
A queste esigenze fa fronte MatrixKnob.VI:

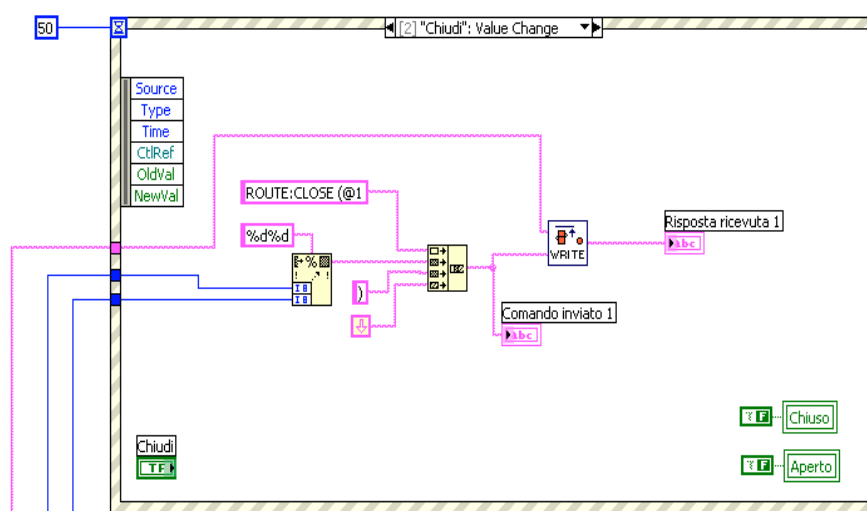
- Lo stato del circuito viene letto all’accensione e riletto continuamente per darne un riscontro visivo in un pannello di Led, e mantenere sempre il bus impegnato in comunicazioni e conferme continue, a conferma che il sistema sia funzionante e pronto a rispondere;
- Gli azionamenti avvengono tramite due manopole, a selezionare riga e colonna da collegare, e due pulsanti, etichettati intuitivamente chiudi e apri;
- Si ha immediato riscontro visivo dell’invio del comando, con lo spegnimento di entrambi i Led “Aperto” e “Chiuso” che indicano lo stato attuale del circuito;
- Solo successivamente al ricevimento della risposta dalla macchina sullo stato attuale, verrà acceso il led “Aperto” o “Chiuso” a conferma che l’azionamento sia stato eseguito;
- Didatticamente vengono mostrati a schermo tutti i comandi inviati e ricevuti.

La conferma dell’azionamento è fondamentale nel caso dell’utilizzo in remoto, senza riscontro visivo, del circuito.

Block Diagram di MatrixKnob.VI

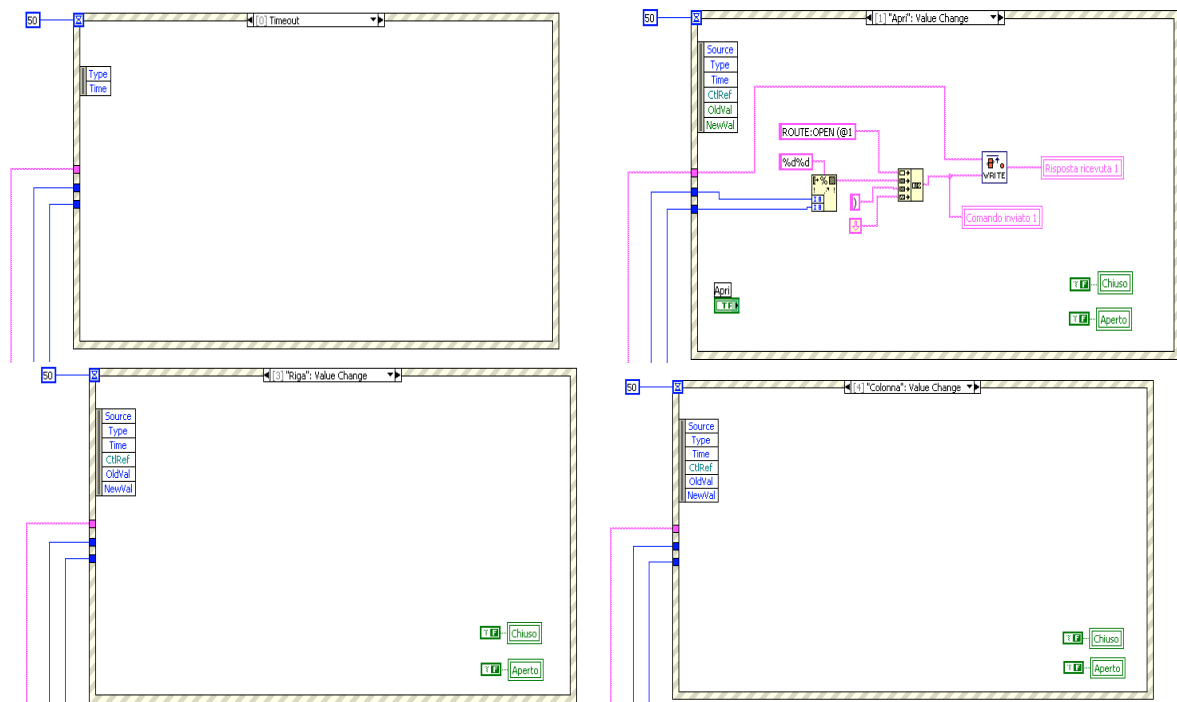
Il Block Diagram è piuttosto complesso. Ne descrivo le funzionalità un pezzo per volta.

Query sullo stato attuale

MatrixWrite viene invocato con la richiesta delle query ROUTE:CLOSE?. La risposta viene scorporata dei caratteri iniziali e ne viene conservata solo la sequenza di 16 caratteri pari a “1” o “0” a seconda se i relais siano chiusi o aperti. La risposta viene anche inviata così come è verso l’alto, verso la sezione del block diagram che accenderà i led “aperto” o “chiuso”.

Analisi della stringa formata da “0” e “1”.

La stringa formata da “0” e “1” viene elaborata tramite la struttura in figura, che accende o spegne i led corrispondenti del pannello a led 4x4.

Accensione led “Chiuso” o “Aperto”

La posizione delle manopole di selezione riga e colonna viene inviata verso la struttura di gestione eventi sovrastante (fili blu che escono dalla sinistra). Il valore di tale manopole viene opportunamente trattato matematicamente a selezionare il carattere di posto corretto nella stringa di risposta alla query (filo fucsia che entra da sinistra). Il confronto con “0” o “1” del carattere selezionato nella risposta accenderà opportunamente uno dei due led “Chiuso” o “Aperto”.

Gestore dell’evento “Bottone Chiudi”

In risposta all’evento di pressione del bottone “Chiudi”, viene generato il comando ROUTE:CLOSE opportuno e inviato allo strumento, in base alla posizione corrente delle

manopole che entra con un tunnel nel gestore degli eventi. L’altro tunnel porta invece l’identificatore della porta seriale utilizzata.

Gli altri gestori di eventi

Altri 4 gestori di eventi sono stati programmati:

- Pressione del bottone “Apri”, il cui codice è analogo a quello relativo alla pressione del bottone “Chiudi”. Le differenze sono che utilizza due variabili locali per l’accesso agli indicatori “Comando Inviato 1” “Risposta Ricevuta 1”, poiché i relativi indicatori erano già stati utilizzati nel gestore di eventi del bottone “Chiudi”. L’altra, ovvia, differenza, è che invia il comando “ROUTE:OPEN” anziché “ROUTE:CLOSE”.
- Timeout. Il gestore di eventi per il time-out che è vuoto. Tale gestore è presente perché si desidera ripetere continuamente le query sullo stato attuale, anche se l’utente non aziona bottoni del front panel.
- Due gestori di eventi per la rotazione delle manopole: vengono immediatamente spenti i due led di indicazione stato attuale del collegamento, in attesa che la query successiva ne rilegga il valore aggiornato e li riaccenda.

Come si vede i costrutti disponibili in LabView sono potenti e permettono di risolvere qualsiasi problema di programmazione, rendendolo di fatto un linguaggio general-purpose.

Inoltre un programma ben scritto in LabView consente a chiunque di utilizzare proficuamente il circuito realizzato in questo tema d’anno.

Capitolo 6 – Conclusioni

Il circuito realizzato rispetta tutte le specifiche iniziali, e soddisfa in pieno gli scopi per il quale è stato realizzato. E' in grado di effettuare l'auto-test, ha un ampio range di tensioni di ingresso, ha superato i test di affidabilità, è modulare ed espandibile, è compatibile con il protocollo SCPI, standard industriale IEEE 488.2, consentendo una facile migrazione del software di controllo da e verso prodotti commerciali equivalenti.
Rispetto a questi ultimi, presenta i vantaggi di un costo inferiore di un fattore almeno 10, una documentazione esaustiva su tutti gli aspetti, caratteristica ottima in un contesto didattico/universitario e tale da consentire la riparazione immediata o lo sviluppo di versioni successive migliorate, e ingombri più contenuti, che consentono di trasportare il sistema, inizialmente pensato per essere cablato permanentemente alla strumentazione del laboratorio.

Valigetta (a doppia apertura) con PC portatile, programmatore, controller, modulo, cavi di collegamento.
L'intero sistema può essere trasportato facilmente sul campo per l'utilizzo o per dimostrazioni didattiche.

In una valigetta possono trovare posto un PC portatile, il sistema Relais Matrix, e un oscilloscopio ATE palmare: è così possibile portare sul campo un intero laboratorio ATE in miniatura, comprese le matrici di switching.

Il sistema finale, punto di arrivo del tema d'anno, è ulteriormente migliorabile. Viene immediato pensare di costruire ulteriori moduli di espansione, identici a quello realizzato, o con topologia di connessione dei relais differente (ad esempio multiplexer anziché matrici),

per potenziare senza sforzo il numero di connessioni automatiche gestibile, e verificare che non ci siano bug nel firmware, che è stato testato con un solo modulo connesso, anche se ne può gestire fino a 4, nella versione attuale, e anche di più con una riprogrammazione opportuna. Il lavoro di sviluppo del firmware può continuare con l'implementazione delle routine di salvataggio/richiamo memorie, progettate diffusamente ma non codificate per mancanza di tempo, e con l'introduzione di ulteriori comandi, se ne appare la necessità.
Il circuito verrà certamente utilizzato in laboratorio per effettuare in automatico tutte le misure che richiedono una riconfigurazione del cablaggio del circuito, effettuate fino ad oggi in manuale, per l'assenza di matrici di switching.

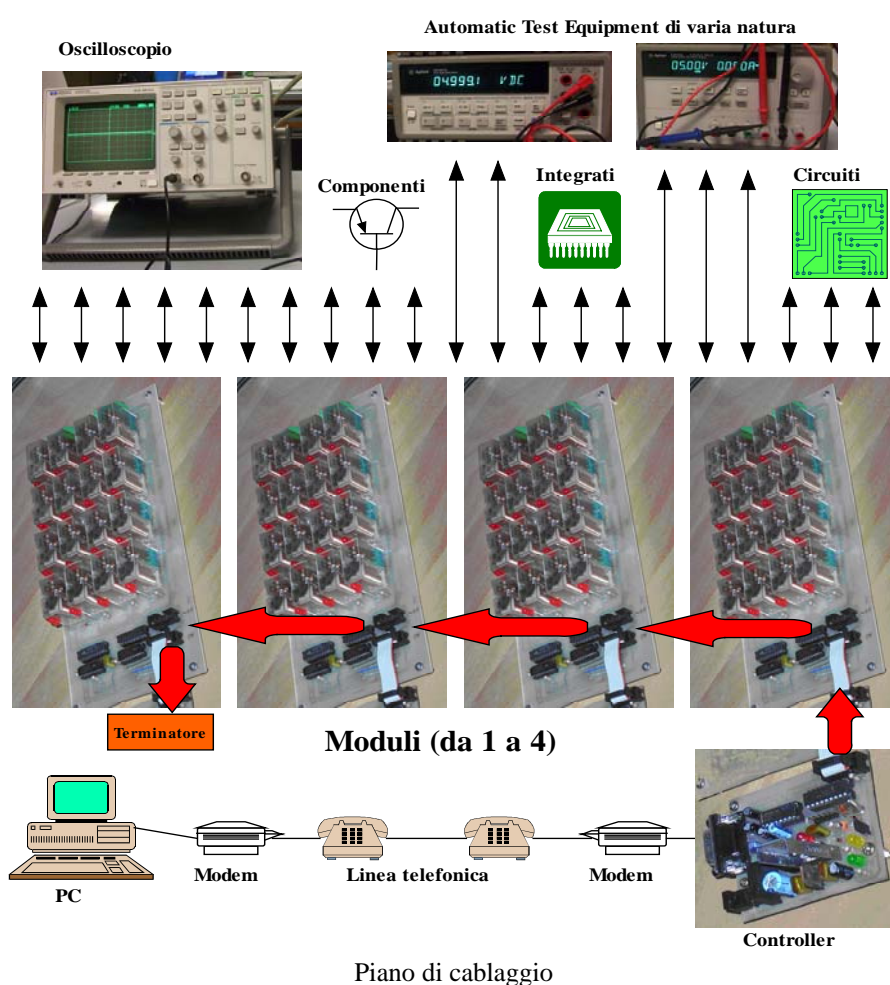
Una volta verificata l'adeguatezza del circuito, se ne possono estendere le potenzialità utilizzando altri bus di comunicazione al posto della porta seriale. Si può pensare ad un protocollo proprietario basato su TCP/IP, in modo da utilizzare il sistema con una connessione Ethernet, o di realizzare un WebServer embedded, o su PC di appoggio, che consenta di accedere all'intero laboratorio in remoto, da qualunque PC connesso a Internet. Molti degli apparecchi presenti in laboratorio, infatti, sono già dotati di un WebServer integrato, con tanto di pannello di controllo virtuale remoto.

Appendice A - Annotazioni pratiche per l’uso

In questa appendice vengono dati consigli pratici per mettere il sistema, conservato in una teca del laboratorio assieme ad una stampa di questa relazione e ad un CD con il software di utilizzo, in grado di funzionare in pochi minuti, anche se sono trascorsi anni dall’ultimo utilizzo, evitando quei comportamenti che potrebbero danneggiarlo.

Cablaggio

Fare riferimento all’introduzione per quel che riguarda il piano di cablaggio del sistema e le specifiche elettriche e di ingombro.

Piano di cablaggio

Il cuore del sistema è la basetta del controller (la più piccola di dimensioni).

Il controller si collega ad una porta seriale disponibile su un PC mediante un cavo seriale null-modem, che contenga, al minimo, la connessione tra i pin 5 di massa, e tra i pin 2 e 3, invertiti, delle due porte seriali. Per indicazioni più precise sulla costituzione del cavo si veda il capitolo 1.

Il circuito si alimenta tipicamente con il *wall-cube* fornito a corredo.

AC/AC Adapter fornito a corredo

In mancanza di questo, si può usare un qualunque alimentatore con tensione d'uscita compresa tra 9 e 15V AC o tra 12 e 18V DC, qualunque polarità. L'assorbimento tipico è di 1 A. Si veda il capitolo 2 per indicazioni precise sull'assorbimento. Il connettore di alimentazione è il popolare 2,5x5,5x9,5 mm.
Aspettare ancora prima di dare tensione al circuito.
Utilizzare un cavo piatto a 10 poli per connettere i moduli in cascata. Su ciascun modulo sono riportate le diciture "in" e "out" accanto ai due connettori per il bus, per distinguere tra l'ingresso e l'uscita. Inserire il ponticello di terminazione catena sull'ultimo modulo collegato. Tale ponticello è presente tra i due header di ingresso e uscita bus di sistema. Il connettore per il cavo piatto utilizzato presenta uno smusso che impedisce di collegarlo capovolto, quindi non dovrebbe essere possibile sbagliare connessione.

IMPORTANTE! Costruzione di un cavo piatto

Se si realizza un cavo piatto a misura, assicurarsi che la banda rossa, presente da un lato del cavo, corrisponda al pin 1 (indicato con una freccia sul connettore) per entrambi i connettori grimpati sul cavo, in modo da realizzare un cavo che effettui il collegamento corretto 1-a-1 di tutti i segnali, e non un cavo che li inverta tutti, con la probabile successiva rottura del circuito.

IMPORTANTE! Non collegare moduli a controller acceso

Per aggiungere o rimuovere moduli dalla catena procedere in questo modo: togliere l'alimentazione al controller, realizzare la nuova catena di moduli, e alimentare nuovamente il controller. Non collegare moduli a caldo poiché i relais potrebbero scattare in uno stato casuale, e il controller non rendersi conto dell'avvenuta variazione del numero di moduli connessi, con risultati imprevedibili.

IMPORTANTE! Ponticello non documentato

Una caratteristica non documentata a dovere, ma solo citata, nella documentazione, è l'header con ponticello presente accanto al quarzo sulla basetta del controller: tale ponticello deve essere inserito per l'uso normale del circuito, e va disinserito solo per le fasi di riprogrammazione del firmware tramite l'header ICSP. Assicurarsi dunque che sia presente il ponticello in tale header. In caso di assenza, alimentando il circuito, il buzzer emetterà una nota continua. Spegnere il circuito, inserire il ponticello, riaccendere il circuito.

IMPORTANTE! Header ISCP

L'header ISCP (a 5 poli) si utilizza durante la riprogrammazione in-circuit del microcontrollore. Non inserire ponticelli in questo header, poiché produrrebbero corti distruttivi. Collegando un programmatore differente dal Fiser's programmer a tale header, controllare tutti i segnali sullo schema elettrico e schema di sbroglio della basetta, poiché differenti programmatori utilizzano differenti piedinature per l'header ICSP. Anche utilizzano il Fiser's programmer, attenzione a non inserire il connettore, che non ha uno smusso di riferimento, capovolto.

LabView

Caricare su un PC su cui sia installato LabView MatrixDemo.VI, presente sul CD allegato al circuito. MatrixDemo.VI rappresenta la maniera più facile in assoluto di controllare il circuito. Selezionare la porta di comunicazione utilizzata, ma non lanciare ancora il VI.
Alimentare il circuito, verificare che i led verde e giallo di alimentazione presente siano accesi, quello rosso di segnalazione errori spento, e che il circuito non emetta alcun "beep", una volta alimentato, a segnalare una condizione di errore hardware per il fallimento della routine di autotest.
Lanciare il VI sul PC premendo il tasto "Run" in Labview. Operando sugli interruttori del pannello virtuale, dovrebbero accendersi e spegnersi i relativi relais sul modulo del primo slot.

In caso di problemi

Se così non fosse, chiudere LabView, e passare all'utilizzo in interattiva con il Terminale di Windows, per capire le cause del malfunzionamento. La porta seriale di comunicazione va configurata per 8 bit di dati, nessuna parità, 1 bit di start, 1 bit di stop, velocità 57600 baud, nessun controllo di flusso. Configurare il Terminale per inviare CR+LF a fine riga. E' una buona idea, a Terminale lanciato e connesso alla porta, spegnere e riaccendere il circuito per

osservare il messaggio di benvenuto *02 Hallo* o un messaggio di errore hardware. A questo punto utilizzare le combinazioni di debug CTRL-E e CTRL-F in modo da poter inserire i successivi comandi osservando l'echo dei caratteri digitati (a conferma che la UART del PIC stia funzionando) e premendo semplicemente INVIO a fine riga (anziché CTRL-H per inviare il new-line).

Utilizzo produttivo

Utilizzare il proprio ambiente di sviluppo preferito per comunicare con la porta seriale, ad esempio il Matlab con gli script di interazione forniti a corredo, oppure un altro linguaggio di programmazione con le proprie librerie.

Utilizzare gli specchietti riassuntivi dei comandi disponibili e degli errori generabili presenti nel capitolo 3 come riferimento rapido durante la programmazione.

Utilizzare il capitolo 5 come riferimento per gli script Matlab e i programmi Labview e come spunto per realizzare le routine di controllo in altri linguaggi di programmazione.

Una volta che i relais scattano come desiderato, collegare strumentazione ATE e dispositivi in test tramite i morsetti a vite presenti sul lato superiore dei moduli. Specie nel caso di utilizzo di alimentatori programmabili, prestare estrema cura a non mettere le relative uscite in corto tramite la matrice di relais.

Approfondimento per utilizzo avanzato. Pause nell'invio di comandi sulla seriale

E' buona norma, durante l'utilizzo, leggere le risposte date dal circuito ai comandi inviati, in modo da ottenere conferma che il comando sia stato eseguito correttamente. Non è previsto alcun controllo di flusso sulla seriale, ma questo non dovrebbe procurare problemi, poiché il protocollo di comunicazione è stato progettato tenendo conto di questa scelta costruttiva.

L'unica accortezza richiesta al programmatore al quale non interessi la velocità di esecuzione dei comandi, e che non voglia incorrere in errori di comunicazione, è quella di inviare un comando per volta, concluso correttamente dal new-line, e di attendere 112ms (caso peggiore) prima di inviare il comando successivo (ovviamente il PC può fare altro durante questo tempo di attesa).

Volendo inviare comandi più rapidamente, una procedura certamente funzionante è quella di attendere solo 12ms[70] dopo l'invio del comando, leggere la riposta del microcontrollore, e

[70] Il tempo, leggermente abbondante, per inviare la stringa di risposta più lunga (caso peggiore), vale a dire la risposta alle query ROUTE:CLOSE? e ROUTE:OPEN? nel caso di tutti e 4 gli slot occupati.

solo in caso di errore attendere ulteriori 100ms prima di inviare il comando successivo. In caso di risposta *01 Completed*, è quindi possibile inviare comandi al ritmo di uno ogni 12ms.

Se si ha confidenza che i comandi inviati siano sempre, o quasi sempre, tutti validi, e si è abituati a non ricorrere a tecniche di programmazione più sofisticate se non strettamente necessarie, si può semplicemente ignorare ogni problema di controllo di flusso e ritardo, e inviare semplicemente i comandi uno di seguito all'altro, rapidamente quanto si desidera, poiché spesso la UART del PC, in configurazione di default, rallenta le trasmissioni quando riceve qualcosa, proprio per venire incontro a dispositivi half-duplex. Sarà la UART stessa, il suo driver, o i layer software dell'ambiente di sviluppo a farsi carico di bufferizzare e inviare lentamente i comandi inviati di seguito dal programmatore disattento. Questo approccio funziona nel 90% dei casi, specie quando è possibile dichiarare esplicitamente il carattere di fine-riga al proprio ambiente di sviluppo.

Procedendo in questo modo, se non si verificano errori, non ci sono problemi. Se si verifica un errore, il circuito emetterà un beep e probabilmente si genererà una cascata di due-tre errori di buffer overrun, uno ogni 100ms, fino a che cessa la trasmissione di caratteri dal PC. Si viene quindi comunque avvisati della perdita di caratteri successivi avvenuta in seguito al primo errore, se non dalla rilettura da programma del messaggio di errore generato dal circuito, almeno dall'emissione dei "beep" di avvertimento direttamente dal buzzer presente sulla basetta. Si ricorda che oltre al buzzer, sul circuito è presente un led di segnalazione. La condizione di led rosso fisso acceso indica che il parser non sta processando l'input fino alla ricezione del successivo new-line.

Ciascun utente, che programmerà il PC, può scegliere l'approccio a lui più congeniale nell'utilizzare la macchina, che si spera possa essere usata proficuamente e con soddisfazione nelle attività di sperimentazione.

Appendice B - Listati dei programmi

Firmware in Assembler

psifthen.inc

```
;
; psifthen 1.2
; Copyright (C) 2000-2003 Paolo Sancono
; http://www.ideegeniali.it/
;
; This program is free software; you can redistribute it and/or modify
; it under the terms of the GNU General Public License as published by
; the Free Software Foundation; either version 2 of the License, or
; (at your option) any later version.
;
; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
; GNU General Public License for more details.
;
; You should have received a copy of the GNU General Public License
; along with this program; if not, write to the Free Software
; Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
;
; Macro in assembler per PIC rendere agevole il controllo di flusso
; vengono implementati if-then if-then-else for-donwto-next repeat-until
; Nel file .asm che intende utilizzare queste macro,
; mettere #INCLUDE <psifthen.inc> subito dopo #INCLUDE <processore.inc>
;
; Esempio
;        LIST      p=16f628
;        #INCLUDE <p16f628.inc>
;        #INCLUDE <psifthen.inc>
;
; Autor       : Paolo Sancono
; Website     : http://www.ideegeniali.it/
; Email       : Changes often - see website
; Version     : 1.0
; Licenza     : GPL V2.0
; Compiler    : MPASM 03.00
; IDE         : MPLAB IDE for Windows/16 V 5.50.00
; Disclaimer  : L'autore non è responsabile di nessun danno diretto o indiretto provocato dall'uso
;               delle macro, per le quali non si garantisce nessuna rispondenza agli scopi previsti.
;               Chi usa queste macro lo fa a suo rischio e pericolo.
; Licenza     : Puoi usare le macro. Puoi anche copiarle, modificarle e ridistribuirle gratuitamente
;               a patto di conservare l'informazione sull'autore originale, sul suo sito internet,
;               tutto questo paragrafo, e distribuirle con la stessa licenza. Non puoi far soldi
;               dalla vendita o uso delle macro. Vedere la General Public License per maggiori info.
;

; -----------------------------------------
; ----- disable those useless warnings -----
; -----------------------------------------

;                  errorlevel -305    ;Using default destination of 1 (file).
                   errorlevel -224    ;Use of this instruction is not recommended (TRIS)
                   errorlevel -302    ;Register in operand not in bank 0.
                                      ;Ensure that bank bits are correct.

; ------------------------
; ----- IF-THEN-ELSE -----
; ------------------------
; Vengono confrontati due file (locazioni RAM) tra loro, oppure un file
; con un literal o con il contenuto dell'accumulatore.
; Il controllo di maggioranza è del tipo maggiore-uguale
; il controllo di minoranza è del tipo minore stretto
; Scambiare di posto i due file da confrontare oppure incrementare o
; decrementare opportunamente di 1 se si ha necessità dell'altro comportamento.
; Occhio ai roll-over oltre gli 8 bit. Questi script considereranno i numeri
; binari a 8 bit senza segno. Se si vuole considerarli con segno, aggiungere
; o togliere .128 a entrambi prima di confrontarli
; Occorre specificare una etichetta per ogni ramo, consigliato usare identificatori
; numerici del tipo iXXX e la lista di simboli per verificare quali sono ancora liberi

; Esempio di utilizzo if-then
;        ifne     Data1,Data2,myif
;                 <istruzioni>
;                 <istruzioni>
;        ifend    myif
;        <resto del programma>

; Esempio di utilizzo if-then-else
```

```
;         ifee      Data1,Data2,myif1
;                   <istruzioni ramo then>
;                   <istruzioni ramo then>
;         ifelse    myif1,myif2
;                   <istruzioni ramo else>
;                   <istruzioni ramo else>
;         ifend     myif2

; Esempio di utilizzo if (condizione) and (condizione) then
;         ifee      data1,data2,myif
;         ifee      data3,data4,myif
;                   <istruzioni>
;                   <istruzioni>
;         ifend     myif

; Esempio di utilizzo if (condizione) or (condizione) then
;         ifee      data1,data2,x1x
;                   goto xxxORxxx
;         ifend     x1x
;         ifee      data3,data4,x2x
;                   goto xxxORxxx
;         ifend     x2x
;         ifelse    xxxORxxx,yyy
;                   <istruzioni>
;                   <istruzioni>
;         ifend     yyy

; Esempio di utilizzo di if(/condizione) goto
;         ifee      data1,data2,label           ; Il salto verrà effettuato per la condizione opposta: ne

ifee      macro     file1,file2,alabel ;if egual (File-File)
          movf      file1,W
          subwf     file2,W
          btfss     STATUS,Z
          goto      alabel
          endm

ifeew     macro     file,alabel                 ;if egual (File-W)
          subwf     file,W
          btfss     STATUS,Z
          goto      alabel
          endm

ifeel     macro     file,literal,alabel;if egual (File-Literal)
          movlw     literal
          ifeew     file,alabel
          endm

ifeelw    macro     literal,alabel              ;if egual (Literal-W)
          sublw     literal
          btfss     STATUS,Z
          goto      alabel
          endm

ifne      macro     file1,file2,alabel ;if not egual (File-File)
          movf      file1,W
          subwf     file2,W
          btfsc     STATUS,Z
          goto      alabel
          endm

ifnew     macro     file,alabel                 ;if not egual (File-W)
          subwf     file,W
          btfsc     STATUS,Z
          goto      alabel
          endm

ifnel     macro     file,literal,alabel;if not egual (File-Literal)
          movlw     literal
          ifnew     file,alabel
          endm

ifnelw    macro     literal,alabel              ;if not egual (Literal-W)
          sublw     literal
          btfsc     STATUS,Z
          goto      alabel
          endm

ifgt      macro     file1,file2,alabel ;if greather than (File-File)
          movf      file1,W
          subwf     file2,W
          btfsc     STATUS,C
          goto      alabel
          endm

ifgte     macro     file1,file2,alabel ;if greather than or egual (File-File)
          movf      file2,W
          ifgtew    file1,alabel
          endm
```

```
ifgtew   macro    file,alabel                ;if greather than or egual (File-W)
         subwf    file,W
         btfss    STATUS,C
         goto     alabel
         endm

ifgtel   macro    file,literal,alabel;if greather than or egual (File-Literal)
         movlw    literal
         ifgtew   file,alabel
         endm

iflt     macro    file1,file2,alabel ;if less than (File-File)
         movf     file2,W
         subwf    file1,W
         btfsc    STATUS,C
         goto     alabel
         endm

ifltw    macro    file,alabel                ;if less than (File-W)
         subwf    file,W
         btfsc    STATUS,C
         goto     alabel
         endm

ifltl    macro    file,literal,alabel;if less than (File-Literal)
         movlw    literal
         ifltw    file,alabel
         endm

ifr      macro    file,min,max,alabel;if file in range (min,max). Estremi compresi.
         movf     file,W                     ;Wraps around 255 (i.e. 240-15 means 240..255 U 0..15):
                                             ;utile per numeri negativi
         ifrw     min,max,alabel             ;Estremi compresi o no? ;DA TESTARE
         endm

ifrw     macro    min,max,alabel             ;if W in range (min,max)    ;DA TESTARE
         addlw    (255 - max) & 0xFF
         addlw    ((max - min) + 1) & 0xFF
         ifbs     STATUS,C,alabel
         endm

ifbs     macro    file,bit,alabel            ;if bit set
         btfss    file,bit
         goto     alabel
         endm

ifbc     macro    file,bit,alabel            ;if bit clear
         btfsc    file,bit
         goto     alabel
         endm

ifelse   macro    alabel1,alabel2
         goto     alabel2
alabel1
         endm

ifend    macro    alabel
alabel
         endm

; ----------------
; ----- DNOP -----
; ----------------
; Esegue due nop con un salto incondizionato alla riga successiva

dnop     macro
         local    label
label    goto     label+1
         endm

; ---------------
; ----- LET -----
; ---------------
; Assegna il contenuto di un file ad un altro
; Nella variante letl assegna un literal ad un file

let      macro    file1,file2
         movf     file2,W
         movwf    file1
         endm

letl     macro    file,literal
         movlw    literal
         movwf    file
         endm

;---------------------------
;----- FOR-DOWNTO-NEXT -----
;---------------------------
;Si conta dal valore iniziale a scendere verso 1
```

```
;Esempio di utilizzo iniziando il conteggio dal valore in W
;         forw      counter,myfor
;         <istruzioni>
;         <istruzioni>
;         next      counter,myfor

;Esempio di utilizzo iniziando il conteggio da un literal
;         forl      counter,.10,myfor
;         <istruzioni>
;         <istruzioni>
;         next      counter,myfor

;Esempio di utilizzo iniziando il conteggio da un file (inizio)
;         for       counter,inizio,myfor
;         <istruzioni>
;         <istruzioni>
;         next      counter,myfor

forw      macro     counter,alabel
          movwf     counter
alabel
          endm

next      macro     counter,alabel
          decfsz    counter,F
          goto      alabel
          endm

for       macro     counter,start,alabel
          movf      start,W
          forw      counter,alabel
          endm

forl      macro     counter,literal,alabel
          movlw     literal
          forw      counter,alabel
          endm

;------------------------
;----- REPEAT-UNTIL -----
;------------------------
;Implementabile facilmente utilizzando gli if che vanno letti "esegui goto se la condizione è falsa"
;Dove ci va scegliere until1 o until2 a seconda se il tipo di if scelto ammette uno o due parametri
;Esempio
;         repeat    myrepeat
;                   <istruzioni>
;                   <istruzioni>
;         until2    ifee,file1,file2,myrepeat

repeat    macro alabel
alabel
          endm

until1    macro     command,parameter,alabel
          command   parameter,alabel
          endm

until2    macro     command,parameter1,parameter2,alabel
          command   parameter1,parameter2,alabel
          endm


; --------------------
; ----- WHILE-DO -----
; --------------------
; Implementabile facilmente utilizzando gli if che vanno letti
; "esegui goto se la condizione è falsa"
; Dopo wile, metti un if con le condizioni negate che hai imparato a usare
; Wile non ha l'h perché while è parola riservata per il compilatore
; Scegliere wile1 o wile2 a seconda se l'if da usare ha 1 o 2 parametri
; Al termine del ciclo, whileend, con le stesse 2 etichette usate per il while
; Esempio
;         wile2     ifee,file1,file2,mywhile1,mywhile2
;                   <istruzioni>
;                   <istruzioni>
;         wilend    mywhile1,mywhile2


wile1     macro     command,parameter,alabel1,alabel2
alabel2   command   parameter,alabel1

wile2     macro     command,parameter1,parameter2,alabel1,alabel2
alabel2   command   parameter1,parameter2,alabel1
          endm

wilend    macro     alabel1,alabel2
          goto      alabel2
alabel1
          endm
```

```
; ------------------
; ----- NegBit -----
; ------------------
; Inverte un bit

negbit   macro    file,bit
         local    wass,wasc,fine
         btfss    file,bit
         goto     wasc
wass     bcf      file,bit
         goto     fine
wasc     bsf      file,bit
fine
         endm

; ---------------------------
; ----- EERead, EEWrite -----
; ---------------------------
; Ricordo che per molti PIC gli indirizzi sono a 6 o 7 bit anziché a 8,
; per i PicMicro con soli 64 o 128 byte di EEProm. I MSB non vengono forzati a 0, bisogna
; assicurarsi che lo siano nel software. Le routine sono state scritte per un PIC16F628
; assicurarsi che non debbano essere modificate per altri PicMicro
; EERead legge una locazione e dà il risultato in W. Controllare che return non
;
; distrugga il contenuto di W nel vostro microcontrollore (era un bug Microchip poi risolto)
; All'uscita ci si trova in bank 0.
;
; EEWrite scrive il contenuto di file in location
; All'uscita ci si trova in bank 0.
; All'inizio attende caso mai c'era una scrittura in atto
;
; EEWait fa il polling di EECON1,WR fino a che non si resetta
; ad indicare che l'operazione di scrittura si è conclusa.
;
; Sono scritte come macro: probabilmente andranno incluse in una routine
; per non allungare il codice: le macro vengono infatti reinserite nel
; codice ogni volta che le si richiama. Suggerito usare gli stessi
; nomi, per tali routine, ma con le prime due E in minuscolo anziché in maiuscolo

EERead   macro    location
         banksel  EEADR              ;Bank 1
         movlw    location ;Setta l'indirizzo a cui leggere
         movwf    EEADR              ;
         bsf      EECON1,RD ;Inizia la fase di lettura
         movf     EEDATA,W ;Restituisce la lettura in W
         banksel  0                  ;Bank 0
         endm

EEWait   macro
         local    loop
         banksel  EECON1             ;Bank 1
loop     btfsc    EECON1,WR ;Aspetta che finisca la write-sequence
         goto     loop
         banksel  0                  ;Bank 0
         endm

EEWrite  macro    location,file      ;File on current bank
         movf     file,W
         EEWait
         banksel  EEDATA             ;Bank 1
         movwf    EEDATA
         bsf      EECON1,WREN        ;Enable write to EEProm
         bcf      INTCON,GIE         ;Disable interrupts
         movlw    0x55               ;Compulsory sequence
         movwf    EECON2             ;
         movlw    0xAA               ;
         movwf    EECON2             ;
         bsf      EECON1,WR ;Inizia write-sequence
         bcf      EECON1,WREN        ;Disable write do EEProm
         bsf      INTCON,GIE         ;Riattiva interrupts
         banksel  0                  ;Bank 0
         endm
```

matrix.asm

```
;
; Relais Matrix PICMicro Firmware 1.0
; Copyright (C) 2005 Paolo Sancono
; http://www.ideegeniali.it/
;
; This program is free software; you can redistribute it and/or modify
; it under the terms of the GNU General Public License as published by
; the Free Software Foundation; either version 2 of the License, or
; (at your option) any later version.
;
; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
; GNU General Public License for more details.
;
; You should have received a copy of the GNU General Public License
; along with this program; if not, write to the Free Software
; Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
;
; Autor       : Paolo Sancono
; Website     : http://www.ideegeniali.it/
; Email       : Changes often - see website
; Version     : 1.0
; Licenza     : GPL V2.0
; Compiler    : MPASM 03.00
; IDE         : MPLAB IDE for Windows/16 V 5.50.00
; Programmer  : Fiser'r Programmer 1.1
; MCU         : PIC16F628-20/P
; Disclaimer  : L'autore non è responsabile di nessun danno diretto o indiretto provocato dall'uso
;               del firmware, per il quale non si garantisce nessuna rispondenza agli scopi previsti.
;               Chi usa questo firmware lo fa a suo rischio e pericolo.
; Licenza     : Puoi usare il firmware. Puoi anche copiarlo, modificarlo e ridistribuirlo gratuitamente
;               a patto di conservare l'informazione sull'autore originale, sul suo sito internet,
;               tutto questo paragrafo, e distribuirlo con la stessa licenza. Non puoi far soldi
;               dalla vendita o uso del firmware. Vedere la General Public License per maggiori info.

; --------------------------------------------------------------------
; ----- Tipo Processore e Configuration word embedded in asm file -----
; --------------------------------------------------------------------

          LIST      p=16f628
          #INCLUDE <p16f628.inc>
          #INCLUDE <psifthen.inc>                    ; Macro utili per cicli for, while e strutture if-then-else

          __CONFIG _BODEN_ON & _CP_OFF & _DATA_CP_OFF & _PWRTE_OFF & _WDT_OFF & _LVP_OFF & _MCLRE_OFF & _HS_OSC

; -------------------------------
; ----- Variabili e Costanti -----
; -------------------------------

          cblock     0x20

;--- Variabili globali
SlotN                ; Numero di slot presenti 0 1 2 3 4
Relais:8             ; 8 byte che memorizzano la configurazione corrente dei 256 relais
                     ; Relais[0] contiene i relais 111 112 113 114 121 122 123 124
                     ;                      nei bit  0   1   2   3   4   5   6   7
                     ; A seguire gli altri.
Lista:8              ; Lista di canali da utilizzare nel comando route:open o route:close
                     ; Le posizioni rappresentate da ciascun bit sono analoghe a quelle di Relais
Ident:8              ; Buffer che contiene la stringa dell'identificatore in parsing che si sta riconoscendo
IdentLength          ; Lunghezza della stringa memorizzata in Ident
LastChar             ; Ultimo carattere letto dalla seriale
Flag                 ; Variabili da 1 bit packed in un byte (sotto i singoli bit)
Errore               ; Errore generato da non inviare subito, ma non appena cessa la ricezione seriale
State                ; Stato della macchina a stati di parsing (sotto le costanti che può assumere questa variabile)
SSDefault            ; Ultimo subsystem utilizzato, che resta come default
SSParsed             ; SubSystem il cui identificatore è in parsing.
                     ; Inizialmente vale SSUndefined per permettere di riconoscere header non validi
                     ; I valori che SSDefault e SSParsed possono assumere sono descritti nelle costanti più sotto.

;--- Variabili temporanee
Looper               ; Programma principale
Temp                 ; Programma principale
sendINDFLooper       ; Routine sendINDF
sendINDFTemp         ; Routine sendINDF
IdentTemp            ; Routine IdentAddChar
ShiftMe              ; Routine shiftout
shiftoutLooper       ; Routine shiftout
sendMsgBcd           ; Routine sendMsg
sendMsgLooper        ; Routine sendMsg
sendMsgTemp          ; Routine sendMsg
autotestLooper       ; Routine autotest
aggiornaModuliLooper ; Routine aggiornaModuli
DelayLooper          ; Routine Delay
DelayCsLooper1       ; Routine DelayCs
DelayCsLooper2       ; Routine DelayCs
          endc

;--- bits packed in byte Flag
FlVerboseAutotest     equ 0      ;Parametro di ingresso per autotest: settato se deve inviare 1/0 sulla seriale
FlErrorPresent        equ 1      ;Settato se c'è un errore da inviare dopo la ricezione della riga corrente
FlLogic               equ 2      ;Usato per i test logici composti (and or tra più condizioni) nel prog principale
FlEcho                equ 3      ;Settato se è richiesto l'echo di ogni carattere ricevuto sulla seriale
FlCrLf                equ 4      ;Settato se è richiesto CR/LF anziché LF in uscita
```

```
FlExclusive          equ 5      ;Settato se è riconosciuto il tag EXCLusive dopo ROUTe:OPEN o ROUTe:CLOSe
                                ;siccome è l'unico tag di 3° livello implementato, si è preferito un Flag di un bit
                                ;anziché una apposita variabile di SubSystem di 3° livello riconosciuto
FlInvert             equ 6      ;Parametro di ingresso routine sendINDF
                                ;Settato complementa relais prima di inviare gli 1 e 0 sulla seriale

           cblock     0x00
;--- Valori possibili per State (stato corrente della macchina a stati di parsing)
StStartOfLine        ; Programma appena avviato, oppure riga precedente terminata e inizio riga successiva
                     ; Cerca "*" o l'inizio di un identificatore ("A".."Z")
StSeekNewLine        ; La linea precedente ha generato un errore, oppure non interessa leggerne la fine,
                     ; si attende la conclusione della riga attuale senza processare nulla
StParseId            ; Si riempie Ident con l'identificatore man mano che ne arrivano i caratteri costituenti.
                     ; L'identificatore verrà riconosciuto non appena si ricevono caratteri consecutivi adeguati.
                     ; Una volta riconosciuto l'identificatore, si setta il SubSystem di default e il SubSystem
                     ; di cui si è appena fatto il parsing. Il comando non verrà eseguito subito, ma a fine riga.
                     ; Si trattano in questo stato anche i duepunti ":", spazi " ", parentesi aperta "(",
                     ; punto interrogativo "?" e il carattere newline che termina il comando corrente
                     ; che verrà eseguito se riconosciuto.
StNewLineAfterQm     ; E' stato riconosciuta una query, dopo il ? si attende per forza newline
                     ; Ricevuto il newline, verrà eseguito il comando e inviati i risultati
StChiocciolaOnly     ; E' stata riconosciuta la parentesi aperta dopo un route:open o route:close, il prossimo
                     ; carattere deve essere per forza una chiocciola @
StRiempiLista        ; Parsing della lista di canali da chiudere/aprire. Si attendono ulteriori numeri di canale,
                     ; virgole "," o la parentesi chiusa ")"
StNewLineAfterParClose; E' stato riconosciuto un ROUTe:OPEN (@ ...) o ROUTe:CLOSe (@ ... )
                     ; Dopo la ) si attende per forza newline. Ricevuto il newline, si esegue il comando
           endc

           cblock     0x00
;--- Valori possibili per SSDefault e SSParsed
SSUndefined          ; Valore predefinito iniziale.
                     ; Se non viene modificato, consente di rilevare un header non riconosciuto
SSCommon             ; Common Commands (Livello 1)
SSSystem             ; System (Livello 1)
SSRoute              ; Route (Livello 1)
SSMemory             ; Memory (Livello 1)
SSClose              ; Close (Livello 2, sotto /Route)
SSOpen               ; Open (Livello 2, sotto /Route)
SSExclusive          ; Exclusive (Livello 3, sotto /Route/Close)
SSCpon               ; Cpon (Livello 2, sotto /System)
SSVer                ; Version (Livello 2, sotto /System)
SSModule             ; Module (Livello 2, sotto /System)
SSState              ; State (Livello 2, sotto /Memory)
SSDelete             ; Delete (Livello 3, sotto /Memory/State)
SSValid              ; Valid (Livello 3, sotto /Memory/State)
SSIdn                ; Idn (Livello 2, sotto /Common)
SSTst                ; Tst (Livello 2, sotto /Common)
SSRst                ; Rst (Livello 2, sotto /Common)
SSSav                ; Sav (Livello 2, sotto /Common)
SSRcl                ; Rcl (Livello 2, sotto /Common)
SSCls                ; Cls (Livello 2, sotto /Common)
           endc

; ----------------------------------------
; ----- Connessioni Hardware del PIC -----
; ----------------------------------------
; Si dichiarano costanti che rispecchiano le connessioni hardware,
; in modo da poter accedere agilmente con mnemonici ai pin del PIC

; Quarzo a 20MHz   Fosc = 20Mhz   Fosc4 = 5Mhz   Tempo di esecuzione NOP = 200 ns

; Mnem : PIN# : DIR : Funzione

; RA0  : 17   :  O  : Shift Register Clock
; RA1  : 18   :  I  : Shift Register Feedback
; RA2  : 1    :  O  : Shift Register Store
; RA3  : 2    :  O  : Shift Register Data
; RA4  : 3    : N/A : N/C
; RA5  : 4    : N/A : ICSP VPP

; RB0  : 6    :  I  : RS232 CTS
; RB1  : 7    :  I  : RS232 Rx
; RB2  : 8    :  O  : RS232 Tx
; RB3  : 9    :  O  : RS232 RTS
; RB4  : 10   :  O  : /Piezo buzzer
; RB5  : 11   :  O  : /Led indicator
; RB6  : 12   : N/A : ICSP PGC
; RB7  : 13   : N/A : ICSP PGD

TrisPortA            equ        B'11110010'
TrisPortB            equ        B'11000011'

ShRegPort            equ        PORTA
ClockBit             equ        0
FeedbackBit          equ        1
StoreBit             equ        2
DataBit              equ        3

CtsPort              equ        PORTB
CtsBit               equ        0
RtsPort              equ        PORTB
RtsBit               equ        3

BuzzPort             equ        PORTB
BuzzBit              equ        4

LedPort              equ        PORTB
LedBit               equ        5

; ----------------------------
; ----- Macro definitions -----
; ----------------------------

qnop       macro                           ; effettua 4 NOP mediante call + return. Occupa 1 sola word in memoria.
```

```
          call       QNOP
          endm

BuzzOn  macro                          ; Seguono banali macro che settano o resettano piedini di uscita del PIC
          bcf        BuzzPort,BuzzBit      ; Accendendo e spegnendo il Led, il Buzzer piezoelettrico
          endm                         ; la linea RTS della seriale

BuzzOff   macro
          bsf        BuzzPort,BuzzBit
          endm

LedOn     macro
          bcf        LedPort,LedBit
          endm

LedOff    macro
          bsf        LedPort,LedBit
          endm

LedToggle macro
          negbit     LedPort,LedBit
          endm

RtsGo     macro
          bsf        RtsPort,RtsBit
          endm

RtsHalt   macro
          bcf        RtsPort,RtsBit
          endm

ifident3 macro       char1,char2,char3,alabel         ; Si usa come gli altri if in psif.inc
          ifeel      IdentLength,.3,alabel            ; Riconosce l'ugualianza tra i primi 3 caratteri di Ident
          ifeel      Ident,char1,alabel               ; e controlla anche che Ident abbia lunghezza 3
          ifeel      Ident+1,char2,alabel
          ifeel      Ident+2,char3,alabel
          endm

ifident4 macro       char1,char2,char3,char4,alabel   ; Come ifident3 ma controlla 4 caratteri
          ifeel      IdentLength,.4,alabel            ; Il codice con macro non è efficiente per quanto riguarda
          ifeel      Ident,char1,alabel               ; l'occupazione in memoria. Se la memoria a disposizione finisce
          ifeel      Ident+1,char2,alabel             ; occorre riscrivere questi controlli utilizzando look-up tables
          ifeel      Ident+2,char3,alabel
          ifeel      Ident+3,char4,alabel
          endm

; -----------------------
; ----- Reset Vector -----
; -----------------------
; Il PIC inizia l'esecuzione delle istruzioni da qui

          ORG        0x0000                ; Reset Vector
          goto       init                  ; Si saltano le definizioni di tavole che è comodo tenere in pagina zero

; ----------------------------------
; ----- Tavole (Look up tables) -----
; ----------------------------------

tavMsgSizes
          bcf        PCLATH,0
          bcf        PCLATH,1
          addwf      PCL,F
          dt         .0,.0                            ; Uno zero in più perché è comodo nella routine che usa la tavola
                                                      ; Un altro zero perché il codice 00 non viene utilizzato
          dt         .9,.5,.13,.14,.12,.17,.17,.17,.17
          dt         .0,.0,.0,.0,.0,.0                ; Le posizioni 0x0A..0x0F non vengono utilizzate: i messaggi sono
                                                      ; memorizzati internamente con indici esadecimali, ma poiché
                                                      ; non sono utilizzati tutti, l'approccio è BCD. All'operatore
                                                      ; esterno gli indici dei messaggi vengono descritti come decimali.
          dt         .11,.17,.13,.13,.33,.16

          IF ((HIGH ($)) != (HIGH (tavMsgSizes)))
                     ERROR "La tavola di salto tavMsgSizes ha superato i confini di pagina"
          ENDIF
          IF ((HIGH ($)) != B'00')
                     ERROR "La tavola di salto tavMsgSizes ha bisogno di ridefinire PCLATH"
          ENDIF

tavMsgText
          bcf        PCLATH,0
          bcf        PCLATH,1
          addwf      PCL,F
          dt         "Completed"                      ;01
          dt         "Hallo"                          ;02
          dt         "Unimplemented"                  ;03
          dt         "Unknown Header"                 ;04
          dt         "Invalid Char"                   ;05
          dt         "Slot out of range"              ;06
          dt         "Chan out of range"              ;07
          dt         "No Module in Slot"              ;08
          dt         "Interrupted chain"              ;09
                                                      ;0A..0F
          dt         "Empty state"                    ;10
          dt         "Different modules"              ;11
          dt         "RS232 Framing"                  ;12
          dt         "RS232 Overrun"                  ;13
          dt         "PaoloSancono,RelaisMatrix,1.0,1.0" ;14
          dt         "Too many modules"               ;15

          IF ((HIGH ($)) != (HIGH (tavMsgText)))
                     ERROR "La tavola di salto tavMsgText ha superato i confini di pagina"
          ENDIF
          IF ((HIGH ($)) != B'00')
                     ERROR "La tavola di salto tavMsgText ha bisogno di ridefinire PCLATH"
          ENDIF
```

```
; -----------------------
; ----- Main Program -----
; -----------------------
; Sezione init eseguita una sola volta
; Sezione receive eseguita in ciclo ad eternum

init
          movlw     0x20                ;Cancella i file register da 0x20 a 0x7f
          movwf     FSR                 ;in modo da assicurare che tutte le variabili
initL     clrf      INDF                ;del programma siano a zero inizialmente
          incf      FSR,F               ;
          btfss     FSR,7               ;
          goto      initL

          movlw     TrisPortA           ;Imposta i TRIS register per le due porte
          tris      PORTA               ;
          movlw     TrisPortB           ;
          tris      PORTB               ;
          movlw     B'00000111'         ;Configura PORTA come Digital I/O (anziché comparatori analogici)
          movwf     CMCON               ;xxxxx111 Comparator disable, RBA0-3 as Digital I/O
          clrf      PORTA               ;Clear port latches
          clrf      PORTB               ;

          bsf       STATUS,RP0          ;Bank 1
          movlw     B'00100110'         ;Configura USART
          movwf     TXSTA               ;Trasmit Status and Control: 8-bit receive, Trasmit enable, Asincronous mode,
                                        ; Baud Rate Generator High Speed (Vs Low Speed)
          movlw     .20                 ;Configura USART
          movwf     SPBRG               ;Baud Rate Generator Register  10: 113636 (115200)  20: 59524 (57600) @20MHz FOsc
          bcf       STATUS,RP0          ;Bank 0
          movlw     B'10010000'         ;Configura USART
          movwf     RCSTA               ;Receive Status and Control: Serial Port Enable, 8-bit receive,
                                        ;Continuous receive enable, Disable Address Detect

          BuzzOff                       ;Spegne cicalino
          LedOff                        ;Spegne Led

          bcf       Flag,FlVerboseAutotest;Determina numero di moduli collegati, con verbose OFF (non stampa "1" o "0")
          call      autotest            ;
          bsf       Flag,FlVerboseAutotest;Verbose On per tutti gli usi successivi (*TST?)

          ifgtel    SlotN,.1,i001       ;Se sono stati trovati tra 1 e 4 moduli
          ifltl     SlotN,.5,i001
                    movlw     0x02      ;02 Hallo
                    call      sendMsg   ;
          ifend     i001      ;Altrimenti all'accensione (e dopo *RST) la stringa di errore stampata da autotest

          call      aggiornaModuli      ;Tutti i relais aperti

nextChar
          ifeel     State,StSeekNewLine,i010      ; Il led viene acceso quando si è nello stato StSeekNewLine
                    LedOn                         ; Viene spento dallo stato StSeekNewLine non appena si riceve newline
          ifend     i010

          ifbc      PIR1,RCIF,i026                ;Se non ci sono altri caratteri da ricevere
                    ifbs      Flag,FlErrorPresent,i027      ;Se c'è un messaggio di errore o informativo in coda
                    ifeel     State,StStartOfLine,i027      ;E siamo nello stato StartOfLine
                                                            ;(la riga precedente è stata processata fino in fondo)
                              movf      Errore,W            ;Lo invia
                              call      sendMsg
                              bcf       Flag,FlErrorPresent ;Resetta il flag di errore presente
                    ifend     i027
                    goto      nextChar                      ;Attende il carattere successivo
          ifend     i026

          ifbs      RCSTA,OERR,i072               ;Test for overrun error (possibile solo se il PC invia un comando,
                                                  ;fa una breve pausa, invia un ulteriore comando mentre il PIC inviava
                                                  ;il suo messaggio di errore.
                    movlw     0x13                ;13 RS232 Overrun
                    call      errore
                    letl      State,StSeekNewLine
                    bcf       RCSTA,CREN          ;Riinizializza la logica di ricezione disattivandola
                    bsf       RCSTA,CREN          ;e riattivandola subito dopo; si è perso (almeno) un byte
          ifend     i072

          ifbs      RCSTA,FERR,i074               ;Test for framing error
                    movlw     0x12                ;12 RS232 Framing
                    call      errore
                    movf      RCREG,W             ;Scarta il byte, perchè contiene dati non validi (errore di trama)
                    letl      State,StSeekNewLine ;Stato: cerca inizio nuova linea
          ifend     i074

          movf      RCREG,W                       ;Legge il carattere successivo
          movwf     LastChar
          ifltl     LastChar,'z'+1,i005           ;Converte in maiuscole le lettere minuscole "a" .. "z"
          ifgtel    LastChar,'a',i005
                    movlw     'A'-'a'
                    addwf     LastChar,F
          ifend     i005

          ifeel     LastChar,0x05,i096            ;Ricevuto CTRL-E (combinazione fuori-standard)
                    negbit    Flag,FlEcho         ;Attiva o disattiva l'echo dei caratteri ricevuti
                    goto      nextChar            ;Utile per il debug al terminale di Windows
          ifend     i096                          ;Rimuovere questa funzione nel firmware definitivo, se desiderato

          ifeel     LastChar,0x06,i100            ;Ricevuto CTRL-F (combinazione fuori-standard)
                    negbit    Flag,FlCrLf         ;Attiva o disattiva il CR accanto al LF in trasmissione
                    goto      nextChar            ;Utile per il debug al terminale di Windows
          ifend     i100                          ;Rimuovere questa funzione nel firmware definitivo, se desiderato

          ifbs      Flag,FlEcho,i097              ;Se è attivo l'echo dei caratteri ricevuti (modalità debug)
                    movf      LastChar,W          ;viene reinviato al PC ogni carattere ricevuto
                    call      trasmit
          ifend     i097
```

```
          ifeel     LastChar,0x0D,i101             ;CTRL-M : Carriage Return
                    goto      nextChar             ;Semplicemente ignorato, così da poter riconoscere CR+LF come
          ifend     i101                                     ;terminatore di riga valido

          ifeel     LastChar,0x03,i109             ;CTRL-C : Break
                    goto      nextChar             ;Semplicemente ignorato, per compatibilità con lo standard SCPI
          ifend     i109

StateCase                               ;Tavola di salto per stato corrente (equivalente a ON ... GOTO del Basic)
          bsf       PCLATH,0            ;La macchina a stati del parser comincia da qui
          bcf       PCLATH,1            ;e processerà il carattere in LastChar (ultimo letto dalla seriale)
          movf      State,W
          addwf     PCL,F
          goto      stStartOfLine
          goto      stSeekNewLine
          goto      stParseId
          goto      stNewLineAfterQm
          goto      stChiocciolaOnly
          goto      stRiempiLista
          goto      stNewLineAfterParClose
          goto      nextChar   ; Should Never come here

          IF ((HIGH ($)) != (HIGH (StateCase)))
                    ERROR "La tavola di salto StateCase ha superato i confini di pagina"
          ENDIF
          IF ((HIGH ($)) != B'01')
                    ERROR "La tavola di salto StateCase ha bisogno di ridefinire PCLATH"
          ENDIF

stStartOfLine
          clrf      SSParsed            ;All'inizio di una nuova riga, si cancellano i SubSystem riconosciuti
          bcf       Flag,FlExclusive    ;precedentemente, a parte il SubSystem di default che rimane valido
          clrf      IdentLength
          ifeel     LastChar,"*",i002   ;Incontrato "*": Subsystem dei common commands, si processa l'header seguente
                    letl      SSDefault,SSCommon
                    letl      State,StParseId
                    goto      nextChar
          ifend     i002

          ifgtel    LastChar,"A",i004   ;Incontrato un carattere alfabetico, si inizia a riempire Ident
          ifltl     LastChar,"Z"+1,i004
                    movf      LastChar,W
                    call      IdentAddChar
                    letl      State,StParseId
                    goto      nextChar
          ifend     i004

          ifeel     LastChar,":",i011   ;Incontrato ":" ad inizio riga: ignorato
                    goto      nextChar
          ifend     i011

          ifeel     LastChar,0x0A,i014  ;Incontrato newline ad inizio riga (riga vuota): ignorato
                    goto      nextChar
          ifend     i014

          movlw     0x05                ;05 Invalid Char      ;Incontrato un carattere diverso: errore!
          call      errore
          letl      State,StSeekNewLine
          goto      nextChar

stParseId
          ifgtel    LastChar,"A",i007   ;if LastChar in ["A".."Z"]  or  LastChar in ["0".."9"]
          ifltl     LastChar,"Z"+1,i007
                    goto      alfanum
          ifend     i007
          ifgtel    LastChar,"0",i017
          ifltl     LastChar,"9"+1,i017
                    goto      alfanum
          ifend     i017
          ifelse    alfanum,notalfanum
                    ; Lo stato ParseId, in caso di caratteri alfanumerici, aggiunge caratteri a Ident
                    ; fino a riconoscere un Header. Quando un header è riconosciuto, si settano le variabili SSDefault,
                    ; SSParsed, Flag(FlExclusive) opportunamente in base a quanto si è riconosciuto. Se si riconosce
                    ; un header non appartenente al SubSystem corrente, si ignora semplicemente il riconoscimento,
                    ; e l'header verrà segnalato come non valido alla fine del comando poiché SSParsed, non aggiornata,
                    ; è rimasta al suo valore iniziale SSUndefined
                    movf      LastChar,W
                    call      IdentAddChar

                    ifident4 "R","O","U","T",i028
                              letl      SSParsed,SSRoute
                              letl      SSDefault,SSRoute
                    ifend     i028

                    ifident4 "C","L","O","S",i029
                    ifeel     SSDefault,SSRoute,i029
                              letl      SSParsed,SSClose
                    ifend     i029

                    ifident4 "O","P","E","N",i030
                    ifeel     SSDefault,SSRoute,i030
                              letl      SSParsed,SSOpen
                    ifend     i030

                    ifident4 "E","X","C","L",i078
                    ifeel     SSDefault,SSRoute,i078
                              ifeel     SSParsed,SSClose,i079
                                        bsf       Flag,FlExclusive
                                        goto      nextChar
                              ifend     i079
                              ifeel     SSParsed,SSOpen,i080
                                        bsf       Flag,FlExclusive
                                        goto      nextChar
                              ifend     i080
```

```
ifend      i078

ifident4 "S","Y","S","T",i033
           letl       SSParsed,SSSystem
           letl       SSDefault,SSSystem
           goto       nextChar
ifend      i033

ifident4 "M","O","D","U",i076
ifeel      SSDefault,SSSystem,i076
           letl       SSParsed,SSModule
           goto       nextChar
ifend      i076

ifident4 "C","P","O","N",i036
ifeel      SSDefault,SSSystem,i036
           letl       SSParsed,SSCpon
           goto       nextChar
ifend      i036

ifident3 "V","E","R",i035
ifeel      SSDefault,SSSystem,i035
           letl       SSParsed,SSVer
           goto       nextChar
ifend      i035

ifident3 "I","D","N",i013
ifeel      SSDefault,SSCommon,i013
           letl       SSParsed,SSIdn
           goto       nextChar
ifend      i013

ifident3 "T","S","T",i015
ifeel      SSDefault,SSCommon,i015
           letl       SSParsed,SSTst
           goto       nextChar
ifend      i015

ifident3 "R","S","T",i023
ifeel      SSDefault,SSCommon,i023
           letl       SSParsed,SSRst
           goto       nextChar
ifend      i023

ifident3 "C","L","S",i031
ifeel      SSDefault,SSCommon,i031
           letl       SSParsed,SSCls
           goto       nextChar
ifend      i031

ifeel      SSDefault,SSSystem,i042
ifeel      SSParsed,SSCpon,i042
; Riconosciuti gli header, si setta opportunamente SSParsed, ma si rimanda l'esecuzione del comando a
; fine riga, così in caso di errori di sintassi nell'istruzione non si esegue nulla.
; Riconosciuto SYSTem:CPON, però, oltre a settare SSParsed opportunamente, si azzerano direttamente
; i bit opportuni in Relais, rimandando l'aggiornamento hardware degli shift register alla fine
; dell'istruzione. Si è evitato così di memorizzare in ulteriori variabili di stato quali Relais
; andavano aperti, ma si può incorrere in errore se dopo SYSTem:CPON XXX si ricevono
; ulteriori caratteri non validi
           ifident3 "A","L","L",i037
                      clrf       Relais
                      clrf       Relais+1
                      clrf       Relais+2
                      clrf       Relais+3
                      clrf       Relais+4
                      clrf       Relais+5
                      clrf       Relais+6
                      clrf       Relais+7
                      goto       nextChar
           ifend      i037
           ifeel      IdentLength,.3,i106
                      ifeel      Ident+1,"0",i044
                      ifeel      Ident+2,"0",i044
                                 ifeel      Ident+0,"1",i046
                                            clrf       Relais
                                            clrf       Relais+1
                                            goto       nextChar
                                 ifend      i046
                                 ifeel      Ident+0,"2",i047
                                            ifltl      SlotN,.2,i071
                                                       movlw      0x08       ;08 No module in slot
                                                       call       errore
                                                       letl       State,StSeekNewLine
                                                       goto       nextChar
                                            ifend      i071
                                            clrf       Relais+2
                                            clrf       Relais+3
                                            goto       nextChar
                                 ifend      i047
                                 ifeel      Ident+0,"3",i048
                                            ifltl      SlotN,.3,i073
                                                       movlw      0x08       ;08 No module in slot
                                                       call       errore
                                                       letl       State,StSeekNewLine
                                                       goto       nextChar
                                            ifend      i073
                                            clrf       Relais+4
                                            clrf       Relais+5
                                            goto       nextChar
                                 ifend      i048
                                 ifeel      Ident+0,"4",i049
                                            ifltl      SlotN,.4,i075
                                                       movlw      0x08       ;08 No module in slot
                                                       call       errore
                                                       letl       State,StSeekNewLine
```

```
                                                                  goto       nextChar
                                                       ifend      i075
                                                       clrf       Relais+6
                                                       clrf       Relais+7
                                                       goto       nextChar
                                            ifend      i049
                                            ifgtel     Ident+0,"5",i102
                                                       movlw      0x06       ;06 Slot out of range
                                                       call       errore
                                                       letl       State,StSeekNewLine
                                                       goto       nextChar
                                            ifend      i102
                                            ifltl      Ident+0,"1",i105
                                                       movlw      0x05       ;04 Unknown Header
                                                       call       errore
                                                       letl       State,StSeekNewLine
                                                       goto       nextChar
                                            ifend      i105
                                 ifend      i044
                                 movlw      0x04       ;04 Unknown Header
                                 call       errore
                                 letl       State,StSeekNewLine
                                 goto       nextChar
                      ifend      i106
           ifend      i042

           goto       nextChar
ifend      notalfanum

ifeel      LastChar,0x0A,i016              ; Incontrato un newline, occorre eseguire il comando riconosciuto
           ifeel      SSDefault,SSCommon,i024
           ifeel      SSParsed,SSRst,i024
                      goto 0                          ; *RST resetta il PIC rieseguendo l'inizializzazione
           ifend      i024

           ifeel      SSDefault,SSCommon,i032
           ifeel      SSParsed,SSCls,i032
                      ;do nothing                     ; *CLS viene riconosciuto ma non sortisce alcun effetto
           ifend      i032

           ifeel      SSDefault,SSSystem,i053
           ifeel      SSParsed,SSCpon,i053
                      call       aggiornaModuli       ; SYSTEM:CPON richiede di aggiornare gli shift-register
                                                      ; i nuovi valori sono già stati memorizzati in Relais
           ifend      i053

           ifeel      SSParsed,SSUndefined,i088       ; Se SSParsed è ancora SSUndefined, messaggio di errore
                      movlw      0x04                 ; 04 Undefined Header
                      call       errore
                      letl       State,StStartOfLine
                      goto       nextChar
           ifend      i088

           movlw      0x01       ;01 Completed        ; Parte comune all'esecuzione dei comandi:
           call       errore                          ; si visualizza 01 Completed e stato successivo inizio-riga
           letl       State,StStartOfLine
           goto       nextChar
ifend      i016

ifeel      LastChar," ",i008               ; Dopo uno spazio si attende l'inizio di un nuovo identificatore
           clrf       IdentLength
           goto nextChar
ifend      i008

ifeel      LastChar,"(",i050               ; Parentesi aperta è carattere valido se in precedenza si è avuto
           bcf        Flag,FlLogic         ; ROUTE:CLOSE o ROUTE:OPEN (e si aspetta la chiocciola)
           ifeel      SSDefault,SSRoute,i051
                      ifeel      SSParsed,SSClose,i052
                                 bsf        Flag,FlLogic
                      ifend      i052
                      ifeel      SSParsed,SSOpen,i056
                                 bsf        Flag,FlLogic
                      ifend      i056
           ifend      i051
           ifbc       Flag,FlLogic,i054               ; Altrimenti messaggio di errore
                      movlw      0x05                 ; 05 Invalid Char
                      call       errore
                      letl       State,StSeekNewLine
                      goto       nextChar
           ifelse     i054,i055
                      letl       State,StChiocciolaOnly
                      goto       nextChar
           ifend      i055
ifend      i050

ifeel      LastChar,":",i009                          ; I due punti separano un header dal successivo
           clrf       IdentLength                     ; occorre pulire Ident e SSParsed per il nuovo header
           ifnel      SSParsed,SSClose,nextChar       ; Se SSParsed è SSClose, non cancella SSParsed,
                                                      ; poiché si può attendere ancora EXCLUSIVE
; La condizione if-not-equal-literal va qui letta goto-if-equal-literal
           clrf       SSParsed
           goto       nextChar
ifend      i009

ifeel      LastChar,"?",i038                          ; Dopo un "?" si attende solo NewLine prima di eseguire la
query
           letl       State,StNewLineAfterQm
           goto       nextChar
ifend      i038

movlw      0x05                                       ; Nessuno dei caratteri precedenti: errore
call       errore                                     ; 05 Invalid Char
letl       State,StSeekNewLine
goto       nextChar
```

```
stChiocciolaOnly
          ifeel      LastChar,"@",i057                          ; In questo stato si attende solo la chiocciola dopo una
                     clrf       Lista                           ; parentesi aperta e si azzerano le strutture di memoria
                     clrf       Lista+1                         ; che ospiteranno la lista di canali
                     clrf       Lista+2
                     clrf       Lista+3
                     clrf       Lista+4
                     clrf       Lista+5
                     clrf       Lista+6
                     clrf       Lista+7
                     clrf       IdentLength
                     letl       State,StRiempiLista
          ifelse     i057,i070                                  ; Se non si riceve una chiocciola, messaggio di errore
                     movlw      0x05                            ; 05 Invalid Char
                     call       errore
                     letl       State,StSeekNewLine
          ifend      i070
          goto       nextChar

stRiempiLista
          ; In questo stato si riempie la variabile Lista man mano che vengono ricevuti numeri di Slot e Canale validi.
          ; Numerosi controlli per errori vengono rilevati, che faranno abortire l'intero comando.
          ; Per questo non si settano direttamente i bit di Relais, ma si prepara a parte la Lista di canali
          ; e si rimanda alla fine della riga l'esecuzione dell'aggiornamento.
          ifgtel     LastChar,"0",i061
          ifltl      LastChar,"9"+1,i061
                     movf       LastChar,W
                     call       IdentAddChar
                     ifgtel     IdentLength,.4,i063
                                movlw      0x05                 ;05 Invalid Char
                                call       errore
                                letl       State,StSeekNewLine
                                goto       nextChar
                     ifend      i063
                     ifeel      IdentLength,.3,i064
                                movf       Ident+0,W  ; Conversione da caratteri ASCII "0" "1" "2" "3" "4" "5"
                                addlw      -("0")-(1) ; a numeri binari a 8 bit       0xFF  0   1   2   3   4
                                movwf      Ident+0
                                movf       Ident+1,W
                                addlw      -("0")-(1)
                                movwf      Ident+1
                                movf       Ident+2,W
                                addlw      -("0")-(1)
                                movwf      Ident+2

                                ifgtel     Ident+0,4,i068
                                           movlw      0x06       ;06 Slot out of range
                                           call       errore
                                           letl       State,StSeekNewLine
                                           goto       nextChar
                                ifend      i068
                                movf       SlotN,W
                                ifgtew     Ident+0,i069
                                           movlw      0x08       ;08 No module in slot
                                           call       errore
                                           letl       State,StSeekNewLine
                                           goto       nextChar
                                ifend      i069
                                ifgtel     Ident+1,4,i087
                                           movlw      0x07       ;07 Chan out of range
                                           call       errore
                                           letl       State,StSeekNewLine
                                           goto       nextChar
                                ifend      i087
                                ifgtel     Ident+2,4,i089
                                           movlw      0x07       ;07 Chan out of range
                                           call       errore
                                           letl       State,StSeekNewLine
                                           goto       nextChar
                                ifend      i089

                                letl       FSR,Lista
                                bcf        STATUS,C   ;Raddoppia Ident+0 in W.  W vale 0 2 4 6 in base a slot number
                                rlf        Ident+0,W
                                addwf      FSR,F                 ;FSR = Lista + 0 2 4 6

                                btfsc      Ident+1,1             ;Se riga "3" o "4" (2 o 3 binario 0-based)
                                incf       FSR,F                 ;FSR = FSR + 1

                                clrf       Temp
                                           incf       Temp,F     ;Temp = B'0001'
                                incf       Ident+2,W
                                forw       Looper,i098
                                           bcf        STATUS,C
                                           rlf        Temp,F
                                next       Looper,i098
                                bcf        STATUS,C
                                rrf        Temp,F                ;Temp = B'0001' B'0010' B'0100' B'1000' in base a Ident+2
                                ifbs       Ident+1,0,i099
                                           swapf      Temp,F     ;Temp = SwapNibbles(Temp) se Ident+1 è riga 2 o riga 4
                                ifend      i099

                                movf       Temp,W                ; Setta il bit in Lista corrispondente allo Slot + Channel
                                iorwf      INDF,F                ; Number riconosciuto nelle righe precedenti
                     ifend      i064
                     goto       nextChar
          ifend      i061

          ifeel      LastChar,":",i058               ; Le liste di canali possono contenere : per intervalli di canali.
                     movlw      0x03                 ; 03 Unimplemented
                     call       errore
                     letl       State,StSeekNewLine
                     goto       nextChar
          ifend      i058
```

```
		ifeel	LastChar,",",i060		; La virgola "," separa un Slot+Channel number dal successivo
			clrf	IdentLength		; Si azzera Ident in modo da utilizzarla nel riconoscimento successivo
			goto	nextChar
		ifend	i060

		ifeel	LastChar,")",i062		; Parentesi chiusa: se segue un newline si esegue il comando
			letl	State,StNewLineAfterParClose
			goto	nextChar
		ifend	i062

		movlw	0x05				; 05 Invalid Char
		call	errore
		letl	State,StSeekNewLine
		goto	nextChar

stNewLineAfterParClose
		; In questo stato ci si aspetta un newline dopo un ROUTE:CLOSE (@...) o ROUTE:OPEN (@...)
		; per eseguire finalmente il comando, utilizzando SSParsed Flag(FlExclusive) e Lista per sapere che tipo
		; di comando è, se il CLOSE è di tipo EXCLUSIVE, e la lista dei canali da Chiudere/Aprire. Se si riceve un altro
		; carattere che non sia newline, è un carattere non valido e si genera l'errore di sintassi
		ifeel	LastChar,0x0A,i045
			bcf	Flag,FlLogic			;FlLogic = (SSParsed == SSOpen) || (SSParsed == SSClose)
			ifeel	SSParsed,SSClose,i092
				bsf	Flag,FlLogic
			ifend	i092
			ifeel	SSParsed,SSOpen,i093
				bsf	Flag,FlLogic
			ifend	i093
			ifeel	SSDefault,SSRoute,i059
			ifbs	Flag,FlLogic,i059
				forl	Looper,.8,i065			; for Looper=0 to 7
					letl	FSR,Lista-1
					movf	Looper,W
					addwf	FSR,F
					let	Temp,INDF
					letl	FSR,Relais-1
					movf	Looper,W
					addwf	FSR,F

					ifeel	SSParsed,SSClose,i094
						ifbs	Flag,FlExclusive,i081
				;Se Exclusive, prima azzera tutti i Relais
							clrf	INDF
						ifend	i081
						movf	Temp,W
				; Alza i bit mascherando i vecchi valori con OR
							iorwf	INDF,F
				; Relais[Looper] = Relais[Looper] OR Lista[Looper]
					ifend	i094
					ifeel	SSParsed,SSOpen,i095
						comf	Temp,W
				; Abbassa i bit mascherando i vecchi valori con AND NOT
						andwf	INDF,F
				; Relais[Looper] = Relais[Looper] AND NOT( Lista[Looper] )
					ifend	i095
				next	Looper,i065
				call	aggiornaModuli
				movlw	0x01	;01 Completed
				call	errore
				letl	State,StStartOfLine
				goto	nextChar
			ifend	i059
		ifend	i045

		movlw	0x05	;05 invalid Char
		call	errore
		letl	State,StSeekNewLine
		goto	nextChar

stNewLineAfterQm
; In questo stato ci si aspetta un newline dopo un "?" qualunque altro carattere è errato
; Ricevuto il newline, si esegue la query. In SSDefault e SSParsed è disponibile il comando da eseguire
; Se SSParsed è rimasto al valore iniziale SSUndefined, è segno che non è stato riconosciuto alcun header valido
		ifeel	LastChar,0x0A,i018
			ifeel	SSDefault,SSCommon,i025
			ifeel	SSParsed,SSIdn,i025
				movlw	0x14	;IDN String;*IDN?
				call	sendString
				call	newline
				letl	State,StStartOfLine
				goto	nextChar
			ifend	i025

			ifeel	SSDefault,SSCommon,i019
			ifeel	SSParsed,SSTst,i019		;*TST?
				call	autotest
				letl	State,StStartOfLine
				goto	nextChar
			ifend	i019

			ifeel	SSDefault,SSSystem,i039
			ifeel	SSParsed,SSVer,i039		;SYSTEM:VERSION?
				movlw	"1"
				call	trasmit
				movlw	"9"
				call	trasmit
				movlw	"9"
				call	trasmit
				movlw	"4"
				call	trasmit
				movlw	"."
				call	trasmit
				movlw	"0"
				call	trasmit
				call	newline
```

```
                        letl      State,StStartOfLine
                        goto      nextChar
              ifend     i039

              ifeel     SSDefault,SSSystem,i077
              ifeel     SSParsed,SSModule,i077          ;SYSTEM:MODULE?
                        movf      SlotN,W
                        addlw     "0"
                        call      trasmit
                        call      newline
                        letl      State,StStartOfLine
                        goto      nextChar
              ifend     i077

              bcf       Flag,FlLogic                    ;FlLogic = (SSParsed == SSOpen) || (SSParsed == SSClose)
              ifeel     SSParsed,SSClose,i107
                        bsf       Flag,FlLogic
                        bcf       Flag,FlInvert
              ifend     i107
              ifeel     SSParsed,SSOpen,i108
                        bsf       Flag,FlLogic
                        bsf       Flag,FlInvert         ;Gli 1 e 0 inviati da sendINDF verranno complementati
              ifend     i108
              ifeel     SSDefault,SSRoute,i082
              ifbs      Flag,FlLogic,i082               ;ROUTE:CLOSE? oppure ROUTE:OPEN?
                        ifeel     SlotN,.0,i083
                                  movlw     "0"
                                  call      trasmit
                                  movlw     "0"
                                  call      trasmit
                                  call      newline
                                  letl      State,StStartOfLine
                                  goto      nextChar
                        ifend     i083
              ; Anziché implementare una routine di moltiplicazione di SlotN per 16, e successiva conversione
              ; da binario a decimale, si è preferito un mero elenco di if-then per ogni possibile valore di SlotN
              ; poiché è questo l'unico caso di utilizzo della conversione binario-decimale. Se ci fossero più
              ; casi, si risparmierebbe più memoria con una routine dedicata. Essendo pochi i casi, si risparmia
              ; memoria con l'approccio ad elenco di if-then.
                        ifeel     SlotN,.1,i084
                                  movlw     "1"
                                  call      trasmit
                                  movlw     "6"
                                  call      trasmit
                        ifend     i084
                        ifeel     SlotN,.2,i085
                                  movlw     "3"
                                  call      trasmit
                                  movlw     "2"
                                  call      trasmit
                        ifend     i085
                        ifeel     SlotN,.3,i086
                                  movlw     "4"
                                  call      trasmit
                                  movlw     "8"
                                  call      trasmit
                        ifend     i086
                        ifeel     SlotN,.4,i090
                                  movlw     "6"
                                  call      trasmit
                                  movlw     "4"
                                  call      trasmit
                        ifend     i090

                        clrf      Looper                ;For Looper = 1 to SlotN
                        repeat    i091                  ;Non si è potuta usare la macro for, perché conta
                        incf      Looper,F              ;a scendere invece a noi interessava contare a salire
                                  letl      FSR,Relais-2          ;FSR = Relais(2*(Looper-1))
                                  bcf       STATUS,C
                                  rlf       Looper,W
                                  addwf     FSR,F
                                  call      sendINDF              ; Invia 8 0 o 1 in base ai bit di INDF
                                  incf      FSR,F
                                  call      sendINDF
                        until2    ifee,Looper,SlotN,i091          ;Fine del ciclo For su Looper

                        call      newline
                        letl      State,StStartOfLine
                        goto      nextChar
              ifend     i082

              movlw     0x04      ;04 Undefined header
              call      errore
              letl      State,StStartOfLine
              goto      nextChar
    ifend     i018

    movlw     0x05      ;05 invalid Char
    call      errore
    letl      State,StSeekNewLine
    goto      nextChar

stSeekNewLine
    ; Si arriva in questo stato se c'è stato un errore nell'istruzione. Non si processa più nulla della riga attuale,
    ; e si attende solo il carattere new-line per iniziare il parsing della nuova riga. Il led rimane acceso quando
    ; si permane in questo stato, e spento quando vi si esce, in modo da avvisare l'operatore che la macchina a stati
    ; non sta processando nulla ma sta solo aspettando il carattere newline.
    ifeel     LastChar,0x0A,i003
              letl      State,StStartOfLine
              LedOff
    ifend     i003
    goto      nextChar

;--------------------
;----- sendINDF -----
```

```
;--------------------
;Invia sulla seriale una sequenza di 8 "0" o "1" in base ai bit del file register puntato da FSR
;Se si setta Flag(FlInvert) verrà invertito ogni bit prima di inviarlo sulla seriale

sendINDF
          let       sendINDFTemp,INDF
          btfsc     Flag,FlInvert
          comf      sendINDFTemp,F
          forl      sendINDFLooper,.8,i104
                    movlw     "0"
                    btfsc     sendINDFTemp,0
                    movlw     "1"
                    call      trasmit
                    rrf       sendINDFTemp,F
          next      sendINDFLooper,i104
          return

;------------------------
;----- IdentAddChar -----
;------------------------
; Ident è un buffer di byte consecutivi in memoria che implementa in Assembler una stringa a lunghezza variabile la cui
; lunghezza attuale viene conservata in IdentLength
; La routine IdentAddChar aggiunge il carattere passato in W in coda a Ident
; Se Ident è già alla dimensione massima, il carattere successivo viene semplicemente ignorato

IdentAddChar
          movwf     IdentTemp
          incf      IdentLength,F
          ifgtel    IdentLength,.9,i006
                    letl      IdentLength,.8
                    return
          ifend     i006
          movf      IdentLength,W
          addlw     Ident-1
          movwf     FSR
          let       INDF,IdentTemp
          return

;--------------------------
;----- aggiornaModuli -----
;--------------------------
; Aggiorna i relais sulle schede con i contenuti attuali della variabile (a 8 byte) Relais,
; inviando i bit di Relais, nell'ordine opportuno, agli shift-register montati sui moduli

aggiornaModuli
          movlw     Relais
          addlw     .7
          movwf     FSR
          forl      aggiornaModuliLooper,.8,i034
                    movf      INDF,W
                    call      shiftout
                    decf      FSR,F
          next      aggiornaModuliLooper,i034
          call      store
          return

;--------------------
;----- autotest -----
;--------------------
; Invia pattern di bit sugli shift register (senza mai mandare store, che aggiornerebbe anche i Latch di uscita)
; siccome i diversi moduli sono collegati in anello, con terminatore sull'ultimo, i bit dopo aver percorso l'anello
; tornano al microcontrollore dal piedino di Feedback.
; autotest verifica quanti moduli sono connessi, conservando l'informazione in SlotN, oppure segnala gli errori di catena
; interrotta o troppi moduli connessi, inviando direttamente l'errore sulla seriale, e settando SlotN a 0 o 5.
; Se Flag[FlVerboseAutotest] è settato, invia 1 o 0 sulla seriale a seconda se l'autotest ha successo o no

autotest
          clrf      SlotN

          bcf       ShRegPort,DataBit                ; Invia 0
          forl      autotestLooper,0xFF,i020         ; agli shift register
                    bcf       ShRegPort,ClockBit     ;
                    bsf       ShRegPort,ClockBit     ;
          next      autotestLooper,i020              ;
          bcf       ShRegPort,ClockBit               ; 257 volte
          bsf       ShRegPort,ClockBit               ;

          ifbs      ShRegPort,FeedbackBit,i022       ; Se trova 1 su Feedback (che ha un resistore di pull-up su scheda)
                    call      autotestFallito
                    movlw     0x09                   ; 09 Interrupted chain
                    call      errore
                    return
          ifend     i022

          forl      autotestLooper,.4,i021           ; Per 4 volte
                    movlw     0xFF                   ; Invia 16 1 sugli shift register
                    call      shiftout
                    call      shiftout
                    incf      SlotN,F
                    ifbc      ShRegPort,FeedbackBit,autotestOk ;Se trova 1, abbiamo determinato il numero di moduli
connessi
          next      autotestLooper,i021                         ; Altrimenti ci sono più moduli

          call      autotestFallito                  ; I moduli connessi sono più di 4
          movlw     0x15                             ; 15 Too many modules
          call      errore

          return

autotestOk
          ifbs      Flag,FlVerboseAutotest,i067
                    movlw     "+"
                    call      trasmit
                    movlw     "1"
                    call      trasmit
```

```
                    call       newline
          ifend     i067
          return

autotestFallito
          ifbs      Flag,FlVerboseAutotest,i066
                    movlw      "+"
                    call       trasmit
                    movlw      "0"
                    call       trasmit
                    call       newline
          ifend     i066
          return

;-------------------------
;----- shiftout store -----
;-------------------------
; Coppia di routine di basso livello per interazione con gli shift-registers
; shiftout
;  Manda gli 8 bit presenti in W agli shift register, dal LST al MSB, con trasmissione sincrona,
;  tramite i pin data e clock
; store
;  Manda un fronte di salita sugli store degli shift register, trasferendo l'informazione
;  dagli shift register ai latch, agli stadi di uscita, ai darlington, ai relais e led

shiftout
          movwf     ShiftMe
          forl      shiftoutLooper,.8,i012
                    bcf        ShRegPort,DataBit
                    btfsc      ShiftMe,0
                    bsf        ShRegPort,DataBit
                    bcf        ShRegPort,ClockBit
                    bsf        ShRegPort,ClockBit
                    rrf        ShiftMe,F
          next      shiftoutLooper,i012
          bcf       ShRegPort,DataBit     ;E' bene che stia sempre a zero, in caso di shift involontari
          return

store
          bcf       ShRegPort,StoreBit
          bsf       ShRegPort,StoreBit
          return

;-------------------------------------------
;----- trasmit newline sendString sendMsg ----
;-------------------------------------------
; Routine di trasmissione caratteri sulla seriale
;
; trasmit
;  Passare in W un byte da trasmettere: verrà trasmesso sulla RS232
;  eventualmente subito dopo aver atteso che terminasse la trasmissione precedente, se ancora in corso
;  in questo caso la procedura è bloccante. Non è stato previsto infatti un buffer di uscita e una gestione
;  ad interrupt, poiché non necessario per il nostro progetto.
;
; newline
;  Invia semplicemente il carattere 0x0A, oppure la coppia 0x0D 0x0A se Flag[FlCrLf] è settato
;
; sendString
;  Invia solo una stringa sulla seriale, senza neppure il carattere new-line dietro.
;  Passare in W il codice BCD del messaggio desiderato (messaggi definiti nelle tavole in pagina 0)
;
; sendMsg
;  Invia un codice numerico a due cifre, uno spazio, un messaggio testuale e un carattere new-line sulla seriale
;  Passare in W, in BCD, l'indice del messaggio da trasmettere. Se l'indice del messaggio è >=3, verrà emesso
;  un beep dal buzzer. Infatti i codici 01 e 02 sono informativi, i codici superiori messaggi di errore

trasmit
          btfss     PIR1,TXIF
          goto      trasmit
          movwf     TXREG
          return

newline
          ifbs      Flag,FlCrLf,i103
                    movlw      0x0D
                    call       trasmit
          ifend     i103
          movlw     0x0A
          call      trasmit
          return

sendMsg
          movwf     sendMsgBcd
          swapf     sendMsgBcd,W
          andlw     0x0F
          addlw     "0"
          call      trasmit
          movf      sendMsgBcd,W
          andlw     0x0F
          addlw     "0"
          call      trasmit
          movlw     " "
          call      trasmit
          movf      sendMsgBcd,W
          call      sendString
          call      newline
          ifgtel    sendMsgBcd,0x03,i043  ;Se il messaggio ha numero 03 o maggiore si emette anche il Beep
                    call       BuzzBeep   ;I messaggi 01 e 02 sono informativi, non di errore
          ifend     i043
          return

sendString
          movwf     sendMsgBcd
          clrf      sendMsgTemp
          for       sendMsgLooper,sendMsgBcd,i040
```

```
                    movf      sendMsgLooper,W
                    call      tavMsgSizes
                    addwf     sendMsgTemp,F
          next      sendMsgLooper,i040
          incf      sendMsgBcd,W
          call      tavMsgSizes
          forw      sendMsgLooper,i041
                    movf      sendMsgTemp,W
                    call      tavMsgText
                    call      trasmit
                    incf      sendMsgTemp,F
          next      sendMsgLooper,i041
          return

; ------------------
; ----- errore -----
; ------------------
; Accoda il messaggio di errore passato in W in Errore (variabile globale)
; Appena cessa la ricezione di caratteri dalla seriale, verrà trasmesso l'errore generato
; Se è già presente un errore, viene conservato il primo e scartati gli ultimi

errore
          btfss     Flag,FlErrorPresent
          movwf     Errore
          bsf       Flag,FlErrorPresent
          return

; --------------------
; ----- BuzzBeep -----
; --------------------
; Effettua un beep accendendo il Buzzer, attendendo 30 centesimi di secondo, e spegnendolo

BuzzBeep
          BuzzOn
          movlw     .10
          call      DelayCs
          BuzzOff
          return

; -----------------------
; ----- Delay, QNOP -----
; -----------------------
; Routine di basso livello che implementano pause bloccanti più o meno lunghe eseguendo il giusto numero di nop
; nel fare i conti per il ritardo si considera anche il tempo di esecuzione di tutte le altre istruzioni, comprese
; call e return per richiamare e ritornare dalla sotto-routine
;
; QNOP       esegue 4 nop (con call e return)
;            QNOP viene richiamata, per semplicità, con la macro qnop anziché con il comando call QNOP
; Delay      esegue W*3+4 (4=2+2= call & return) op's
; Delay100us esegue 500 nop's = 100us, con FOsc=20 MHz
; DelayCs    aspetta W centesimi di secondo (appena abbondanti)
; DelaySec   aspetta un secondo (appena abbondante)

Delay100us movlw     .165
Delay               movwf     DelayLooper
DelayL              decfsz    DelayLooper,F
                    goto      DelayL
QNOP                return

DelaySec   movlw     .100
DelayCs             movwf     DelayCsLooper1
DelayCsL1  decfsz    DelayCsLooper1,F
                    goto      DelayCsC1
                    return
DelayCsC1  movlw     .100
                    movwf     DelayCsLooper2
DelayCsL2  decfsz    DelayCsLooper2,F
                    goto      DelayCsC2
                    goto      DelayCsL1
DelayCsC2  call      Delay100us
                    goto      DelayCsL2

; -------------------------
; ----- END Directive -----
; -------------------------

          END
```

Funzioni Matlab

Funzioni di basso livello

matrixinit.m

```
function MatrixInit(Port);
% MatrixInit
%
% Funzione Matlab che crea la variabile globale MatrixSp, che contiene
% l'handle di un oggetto porta seriale da utilizzare per comunicare con
% lo strumento da laboratorio Relais Matrix. InitMatrix va richiamato in
% interattiva all'inizio della sessione di lavoro.
%
% Sintassi: MatrixInit(Port)
%
% Parametri di ingresso opzionali
%
% Port  :  E' un numero intero da 1 a 8. Indica la porta COM da usare per la
%          comunicazione con la porta seriale. Default = 4

global MatrixSp;

if (~exist('Port'))
    Port = 4;
end
if (round(Port)~=Port)
    disp('Port deve essere un numero intero');
    return
end
if ((Port<1)|(Port>8))
    disp('Port deve essere un intero compreso tra 1 e 8');
    return
end

ComPort=strcat('COM',int2str(Port));
MatrixSp=serial(ComPort,'BaudRate',57600,'DataBits',8,'FlowControl','None', ...
    'DataBits',8,'StopBits',1,'Parity','None','Terminator','LF','Timeout',1);
fopen(MatrixSp);
%In caso di errore questi vengono indicati direttamente da fopen,
%non c'è bisogno di intercettarli e riportarli all'utente

assignin('base','MatrixSp',MatrixSp);
```

matrixdone.m

```
function MatrixDone;
% MatrixDone
%
% Funzione Matlab che distrugge variabile globale MatrixSp, che contiene l'handle
% di un oggetto porta seriale da utilizzare per comunicare con lo strumento da
% laboratorio Relais Matrix. CloseMatrix va richiamato in interattiva alla
% fine della sessione di lavoro
%
% Sintassi: MatrixDone
%
% Nota: MatrixSp deve essere presente nel workspace prima di richiamare
%       questa funzione

global MatrixSp
fclose(MatrixSp);
delete(MatrixSp);
clear MatrixSp;
evalin('base','clear MatrixSp');
```

matrixread.m

```
function Risposta=MatrixRead;
% MatrixRead
%
% Legge una intera riga terminata dal carattere newline dalla porta seriale
```

```
% alla quale è stato collegato il circuito Relais Matrix. Il carattere
% newline non verrà passato in uscita. Se non viene incontrato alcun
% newline, viene passata in uscita una stringa vuota dopo il timeout
% dell'oggetto seriale.
%
% Sintassi: s = MatrixRead
%
% Parametri di uscita
%
%  Risposta  :  Stringa ricevuta dalla porta seriale
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixRead

global MatrixSp;
if (get(MatrixSp,'BytesAvailable')>0)
    Risposta=fscanf(MatrixSp);
end
```

matrixwrite.m

```
function MatrixWrite(cmd);
% MatrixWrite
%
% Invia una stringa terminata dal carattere newline sulla porta seriale
% alla quale è stato collegato il circuito Relais Matrix. Il carattere
% newline non va passato in ingresso.
%
% Sintassi: MatrixWrite(cmd)
%
% Parametri di ingresso obbligatori
%
%  cmd  :  Stringa da inviare sulla porta seriale
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixWrite

global MatrixSp;
fprintf(MatrixSp,cmd);
```

matrixflush.m

```
function MatrixFlush;
% MatrixFlush
%
% Svuota il buffer di lettura dell'oggetto seriale MatrixSp utilizzato
% per comunicare con il circuito Relais Matrix
%
% Sintassi: MatrixFlush
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixWrite

global MatrixSp;
while((get(MatrixSp,'BytesAvailable'))>0)
    junk=fread(MatrixSp,1);
end
```

Funzioni di medio livello

matrixclose.m

```
function MatrixClose(Ch);
% MatrixClose
%
% Invia il comando ROUTE:CLOSE (@ ...) tramite MatrixWrite
```

```
% Il comando CLOSE chiude i relais selezionati mascherando quelli già
% chiusi, che resteranno chiusi. MatrixClose agisce esclusivamente
% sui relais del Modulo collegato allo slot 1 (tipicamente al circuito
% è collegato un solo modulo, al quale verrà assegnato lo slot 1).
%
% Sintassi: MatrixClose(Ch)
%
% Parametri di ingresso opzionali
%
%  Ch  :  Vettore contenente numeri di canale di relais da chiudere sul
%         modulo alloggiato nello slot 1 sul circuito Relais Matrix
%         I relais hanno numeri di canale 11 12 13 14 21 22 23 24
%         31 32 33 34 41 42, così come prevede lo standard industriale.
%         Default = Vettore vuoto: invia il comando con una lista vuota,
%         e questo non sortirà alcun effetto
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixWrite
% NOTA: Non viene controllato se il comando genera errori. Utilizzare MatrixRead.

global MatrixSp;
cmd='ROUTE:CLOSE (@';
while length(Ch)>0
    if (isempty(find(Ch(1) == [11,12,13,14,21,22,23,24,31,32,33,34,41,42,43,44])))
        disp('I numeri di canale devono essere scelti tra i seguenti:');
        disp('11 12 13 14 21 22 23 24 31 32 33 34 41 42 43 44');
        return
    end
    cmd=strcat(cmd,'1',int2str(Ch(1)),',');
    Ch(1)=[];
end
cmd=strcat(cmd,')');

MatrixWrite(cmd);
```

matrixcloseexclusive.m

```
function MatrixCloseExclusive(Ch);
% MatrixCloseExclusive
%
% Invia il comando ROUTE:CLOSE:EXCLUSIVE (@ ...) tramite MatrixWrite
% Il comando CLOSE:EXCLUSIVE chiude i relais selezionati
% aprendo tutti gli altri. MatrixCloseExclusive agisce esclusivamente
% sui relais del Modulo collegato allo slot 1 (tipicamente al circuito
% è collegato un solo modulo, al quale verrà assegnato lo slot 1).
%
% Sintassi: MatrixCloseExclusive(Ch)
%
% Parametri di ingresso opzionali
%
%  Ch  :  Vettore contenente numeri di canale di relais da chiudere sul
%         modulo alloggiato nello slot 1 sul circuito Relais Matrix
%         I relais hanno numeri di canale 11 12 13 14 21 22 23 24
%         31 32 33 34 41 42, così come prevede lo standard industriale.
%         Default = Vettore vuoto: apre tutti i relais senza chiuderne alcuno
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixWrite
% NOTA: Non viene controllato se il comando genera errori. Utilizzare MatrixRead.

global MatrixSp;
cmd='ROUTE:CLOSE:EXCLUSIVE (@';
while length(Ch)>0
    if (isempty(find(Ch(1) == [11,12,13,14,21,22,23,24,31,32,33,34,41,42,43,44])))
        disp('I numeri di canale devono essere scelti tra i seguenti:');
        disp('11 12 13 14 21 22 23 24 31 32 33 34 41 42 43 44');
        return
    end
    cmd=strcat(cmd,'1',int2str(Ch(1)),',');
    Ch(1)=[];
end
cmd=strcat(cmd,')');
```

```
MatrixWrite(cmd);
```

matrixopen.m

```
function MatrixOpen(Ch);
% MatrixOpen
%
% Invia il comando ROUTE:OPEN (@ ...) tramite MatrixWrite
% Il comando OPEN apre i relais selezionati. MatrixOpen agisce esclusivamente
% sui relais del Modulo collegato allo slot 1 (tipicamente al circuito
% è collegato un solo modulo, al quale verrà assegnato lo slot 1).
%
% Sintassi: MatrixOpen(Ch)
%
% Parametri di ingresso opzionali
%
%  Ch  :  Vettore contenente numeri di canale di relais da chiudere sul
%         modulo alloggiato nello slot 1 sul circuito Relais Matrix
%         I relais hanno numeri di canale 11 12 13 14 21 22 23 24
%         31 32 33 34 41 42, così come prevede lo standard industriale.
%         Default = Vettore vuoto: invia il comando con una lista vuota,
%         e questo non sortirà alcun effetto
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixWrite
% NOTA: Non viene controllato se il comando genera errori. Utilizzare MatrixRead.

global MatrixSp;
cmd='ROUTE:OPEN (@';
while length(Ch)>0
    if (isempty(find(Ch(1) == [11,12,13,14,21,22,23,24,31,32,33,34,41,42,43,44])))
        disp('I numeri di canale devono essere scelti tra i seguenti:');
        disp('11 12 13 14 21 22 23 24 31 32 33 34 41 42 43 44');
        return
    end
    cmd=strcat(cmd,'1',int2str(Ch(1)),',');
    Ch(1)=[];
end
cmd=strcat(cmd,')');

MatrixWrite(cmd);
```

Script dimostrativo

pizzica.m

```
function Pizzica;
% Pizzica
%
% Suona la pizzica (danza popolare del salento) mediante RelaisMatrix
%
% Sintassi: Pizzica
%
% NOTA: MatrixSp deve essere già presente come variabile globale,
% e deve contenere l'handle per un oggetto seriale già aperto per
% comunicare con lo strumento. Il modo più facile per ottenere la
% cosa è invocare MatrixInit a riga di comando prima di usare MatrixWrite

MatrixWrite('*RST');
MatrixWrite('ROUTE:CLOSE:EXCLUSIVE (@)');

while (1)
    c('111,112,113,114,,,,,,,,');
    o('111,,,,,,,,,,,,,,,,,,,,');
    o('112,,,,,,,,,,,,,,,,,,,,');
    ce('111,112,,,,,,,,,,,');
    o('111,,,,,,,,,,,,,,,,,,,,');
    o('112,,,,,,,,,,,,,,,,,,,,');

    c('121,122,123,124,,,,,,,,');
```

```
    o('123,,,,,,,,,,,,,,,,,,,,');
    o('124,,,,,,,,,,,,,,,,,,,,');
    ce('123,124,,,,,,,,,,,');
    o('123,,,,,,,,,,,,,,,,,,,,');
    o('124,,,,,,,,,,,,,,,,,,,,');

    c('131,132,133,134,141,142');
    o('131,132,133,,,,,,,,,,,,');
    o('134,141,142,,,,,,,,,,,,');
    c('141,142,143,144,131,132');
    o('141,142,143,,,,,,,,,,,,');
    o('144,131,132,,,,,,,,,,,,');

    c('111,112,113,114,121,122');
    o('121,111,,,,,,,,,,,,,,,,');
    o('122,112,,,,,,,,,,,,,,,,');
    ce('121,122,,,,,,,,,,,');
    o('121,,,,,,,,,,,,,,,,,,,,');
    o('122,,,,,,,,,,,,,,,,,,,,');


    c('131,132,133,134,,,,,,,,');
    o('131,,,,,,,,,,,,,,,,,,,,');
    o('132,,,,,,,,,,,,,,,,,,,,');
    ce('131,132,,,,,,,,,,,');
    o('131,,,,,,,,,,,,,,,,,,,,');
    o('132,,,,,,,,,,,,,,,,,,,,');

    c('141,142,143,144,,,,,,,,');
    o('143,,,,,,,,,,,,,,,,,,,,');
    o('144,,,,,,,,,,,,,,,,,,,,');
    ce('143,144,,,,,,,,,,,');
    o('143,,,,,,,,,,,,,,,,,,,,');
    o('144,,,,,,,,,,,,,,,,,,,,');

    c('111,112,113,114,121,122');
    o('111,112,113,,,,,,,,,,,,');
    o('114,121,122,,,,,,,,,,,,');
    c('121,122,123,124,111,112');
    o('121,122,123,,,,,,,,,,,,');
    o('124,111,112,,,,,,,,,,,,');

    c('131,132,133,134,141,142');
    o('141,131,,,,,,,,,,,,,,,,');
    o('142,132,,,,,,,,,,,,,,,,');
    ce('141,142,,,,,,,,,,,');
    o('141,,,,,,,,,,,,,,,,,,,,');
    o('142,,,,,,,,,,,,,,,,,,,,');
end


function c(list);
    MatrixWrite(strcat('CLOS (@',list,')'));
    pause(0.03);
    MatrixRead;

function ce(list);
    MatrixWrite(strcat('CLOS:EXCL (@',list,')'));
    pause(0.03);
    MatrixRead;

function o(list);
    MatrixWrite(strcat('OPEN (@',list,')'));
    pause(0.03);
    MatrixRead;
```

Appendice C - GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially.
Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or

"History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an
Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
    Copyright (c)  YEAR  YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.2
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendice D - GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent
license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Fonti delle figure, Software utilizzato

Diagrammi e figure relativi alla macchina presa a modello, Agilent 34970A "Data acquisition and switch unit", e al suo modulo accessorio, Agilent 34974A "4x8 Switch Matrix for Agilent 34970A", sono stati ricavati direttamente dai data sheet in formato PDF della Agilent Technologies, scaricati dal rispettivo sito, indicato nella bibliografia.

Diagrammi e figure relativi ai circuiti integrati utilizzati nel progetto (compreso il microcontrollore), sono stati ricavati dai datasheet PDF dei rispettivi produttori, scaricati dai rispettivi siti, indicati nella bibliografia.

Le illustrazioni relative al Fiser's programmer di Fiser sono state scaricate dal sito dell'autore.
http://www.jofi.it/fiser/

I diagrammi con il pinout e il nome dei connettori della seriale sono stati scaricati dal sito "Hardware book"
http://www.hardwarebook.net/

La foto del laboratorio ATE è stata trovata con la funzione di ricerca immagini di altavista.
http://www.av.com/

Le foto del TMRC (Tech Model Railroad Club) sono state scaricate dalla "MIT Gallery Special Exhibits Page"
http://members.aol.com/chopstcks/gallery2/mit/tmrc2.htm

Tutte le altre foto sono state scattate dall'autore con una fotocamera digitale Kodak CX6230.

Tutti gli altri diagrammi sono stati realizzati dall'autore tramite print-screen o funzioni di esportazione immagini mentre sul PC giravano i programmi:

Lorenzo Lutti's FidoCad 0.96b

Piccoli schemi di circuiti elettrici schizzati in nero su bianco.

Cercare su news://it.hobby.elettronica/ una URL aggiornata per il programma

Cadsoft EagleCad 4.14

Schema elettrico e PCB dei circuiti realizzati.

http://www.cadsoft.de/

SmartDraw Professional 6.03

Diagramma con il sistema Relais Matrix completo, Diagramma degli stati del Parser.

http://www.smartdraw.com/

Il computer utilizzato per la realizzazione del sistema, e la relativa documentazione, è un Notebook ECS con processore Transmeta Efficeon.

Altri programmi utilizzati

MathWorks Matlab 6.2

http://www.mathworks.com/

National Instruments LabView 7.0 Student

http://www.ni.com/

Paolo Sancono's Drill-aid 1.5

http://www.ideegeniali.it/

Microsoft Blocco Note 5.1

http://www.microsoft.com/

MicroEngineering Labs EpicWin 2.45

http://www.melabs.com/

Microchip Mpasm assembler 3.00

http://www.microchip.com/

Microchip Mplab IDE 5.50.00

http://www.microchip.com/

HilGraeve/Microsoft Terminale di Windows 5.1

http://www.microsoft.com/

Bibliografia e Webliografia

Informatica

Storia dell'informatica. Informazioni sul Tech Model RailRoad Club e su Richard Stallman, fondatore della Free Software Foundation.
Steven Levy, "Hackers – Gli eroi della rivoluzione informatica", Shake ed. Underground

Scrittura di un parser
Paolo Sancono, "Generazione dell'albero corrispondente ad un documento XML mediante un parser scritto in Java", Tema d'anno per il corso di Sistemi Informativi, Politecnico di Bari, A.A. 2005

Testo della Gnu General Public License (GPL)
http://www.gnu.org/licenses/gpl.txt
Testo della Gnu Free Documentation License (FDL)
http://www.gnu.org/licenses/fdl.txt

Elettronica applicata

Newsgroup in lingua italiana di riferimento per l'elettronica applicata in generale e relativa ai microcontrollori. Sui newsgroup figurano un migliaio di post a mio nome.
news://it.hobby.elettronica/
news://it.hobby.elettronica.digitale/

Newsgroup in lingua inglese di riferimento per l'elettronica applicata
news://sci.electronic.circuits.design/

Schemi di connettori del presente e del passato.
http://www.hardwarebook.net/

Standard RS-232.
http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html
http://www.arcelect.com/rs232.htm
http://www.beyondlogic.org/serial/serial.htm

Assembler dei PIC

PicBook: corso di autoapprendimento all’uso dei microcontrollori PIC della Microchip.

http://www.mikroelektronika.co.yu/english/product/books/PICbook/

Il Picbook è disponibile in lingua inglese, gratuitamente, sul sito indicato, e tradotto in lingua italiana, a pagamento, in libreria.

MpAsm 3.0 User guide

Manuale MpLab 5.5

Data Sheet PIC16F628

http://www.microchip.com/

Data Sheet dei componenti utilizzati

I data sheet degli integrati utilizzati nel circuito sono stati scaricati, in formato PDF, dai siti dei rispettivi produttori.

Agilent Technologies

34970A User's guide english version

34970A Product overview

34970A Quick reference guide

http://www.agilent.com/

Microchip

40300c - PIC16F62X Data Sheet

http://www.microchip.com/

Philips Semiconductors

74HC595 - 8 bit serial-in/serial or parallel-out shift register with output latches, 3-state

http://www.semiconductors.philips.com/

ST / SGS Thomson Microelectronics

ULN2803 - Eight Darlington Array

http://www.st.com/

http://www.sgs-thomson.com/

Dallas Semiconductors / Maxim

MAX232 - +5V-Powered, Multichannel RS-232 Driver Receiver

http://www.maxim-ic.com/

National Semiconductor

LM78XX Series Voltage Regulators

http://www.national.com/